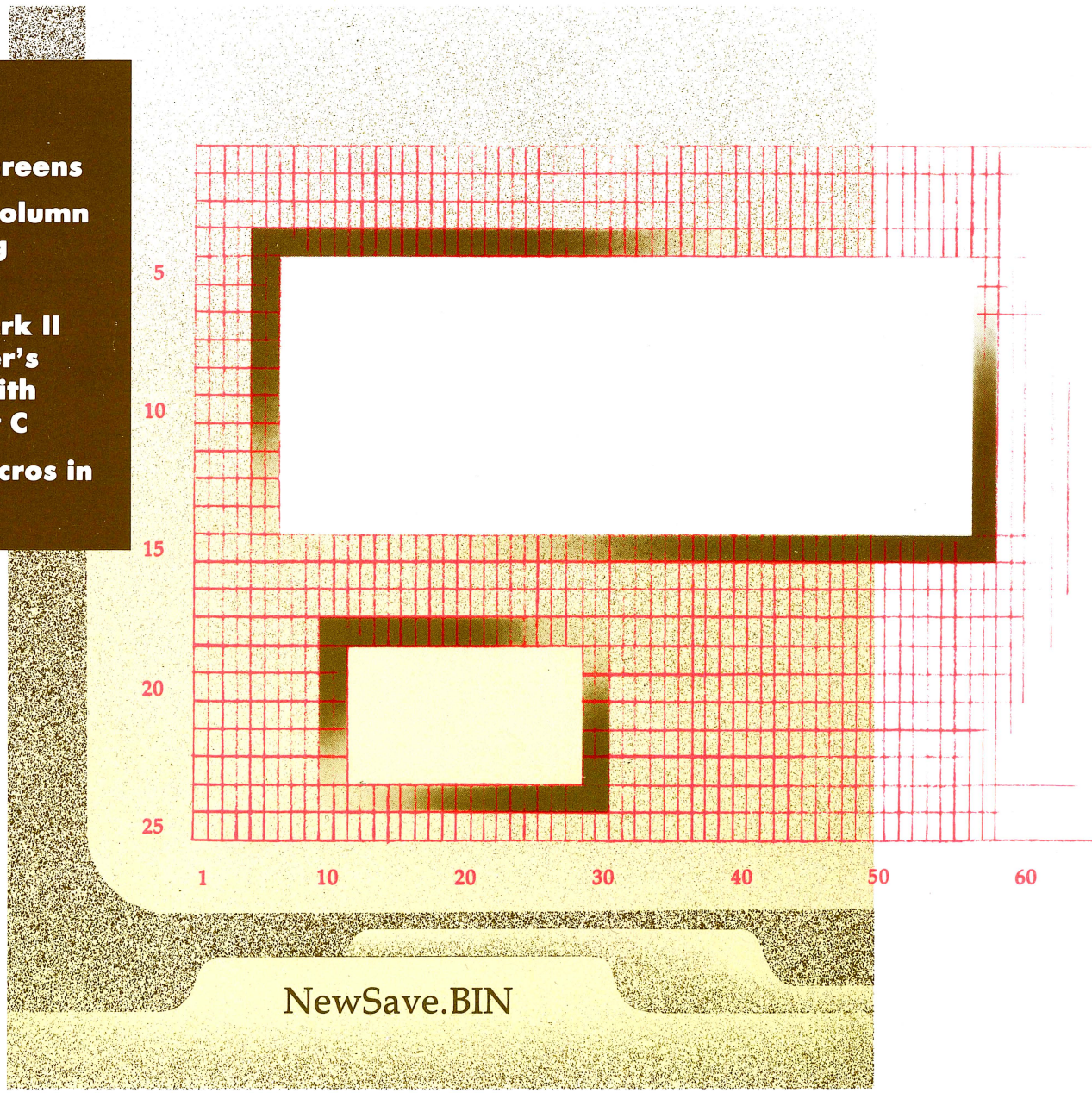
TechNotes

ASHTON-TATE'S TECHNICAL JOURNAL
FEATURING dBASE AND FRAMEWORK

JUNE 1987
$6.00

- **Save and Restore Partial Screens**
- **Snaked Column Reporting**
- **Using the Framework II Developer's Toolkit with Microsoft C**
- **Using Macros in RapidFile**



ASHTON-TATE

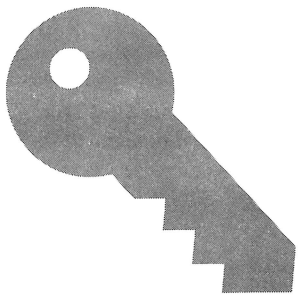
TechNotes

TABLE OF CONTENTS

JUNE 1987

dBASE III PLUS	Save and Restore Partial Screens *by Kenneth N. Getz*	3
	Snaked Column Reporting *by Software Support Technicians*	20
	dBASE Tips	27
Framework II	Using the Framework II Developer's Toolkit with Microsoft C *by Kenneth N. Getz*	30
	Translating FRED Programs from One Language to Another *by Paul Bukauskas*	39
	Framework Tips	42
RapidFile	Automating Applications Using Macros *by Richard J. Augelli*	45
	RapidFile Tips	50
MultiMate	Line and Box Drawings with the HP LaserJet *by Meryl Horowitz*	52
	MultiMate Tips	58
Departments	Read Me First *by Brenda Johnson-Grau*	1
	Announcements: MultiMate Advantage II Debuts Calendar of Training Courses	60 62
	Anomalies	63
	On-Line Services	69
	Corrigenda	76

READ ME FIRST



BY BRENDA JOHNSON-GRAU

June's cover article is a rewritten version of a program that we presented in the January 1987 issue: "Save and Restore Partial Screens," by Kenneth N. Getz. The title says it all: Now you can save and restore partial screens as well as full screens.

Last month saw the release of the *Framework II Developer's Toolkit.* This month we feature "Using the *Framework II Developer's Toolkit with Microsoft C.*" Because all the examples in that product's manuals are written for use with the Lattice C Compiler, users of Microsoft C would have been out of luck—if not for your friendly C programmers at ***TechNotes.***

The technicians at the Software Support Center worked together to fulfill a common request from dBASE III PLUS users for a snaked column custom report program. "Snaked Column Reporting" allows dBASE III PLUS users to create custom reports that wrap like a phone book does. It places the records down the first column, then moves to the top of the second column and continues to place the records, and so on.

MultiMate Advantage II, Ashton-Tate's new word-processing program, is heralded in Announcements. This new version adds over 30 new features, including a dBASE merge function and pull-down menus. This month's MultiMate article is "Line and Box Drawing Using the HP LaserJet," by Meryl Horowitz. It gives step-by-step instructions to using these MultiMate Advantage functions with the HP LaserJet.

Using keystroke macros is the topic of Richard J. Augelli's RapidFile article. "Automating Applications Using Macros," outlines the creation and use of macros in RapidFile and provides a boilerplate text library as a sample application.

As you can see from the Table of Contents, we no longer have a department called Usage Tips. We have divided this department into separate departments by product and placed them with the articles for that product. We hope that this division will help readers find the information for a particular product more quickly.

International has also been dissolved as a department. We plan to continue to include articles submitted by Ashton-Tate's international subsidiaries, but these articles will appear in the particular product category. This month we have a Framework II article, "Translating FRED Programs from One Language to Another," which discusses the basis steps in translating FRED code and gives some English to Spanish examples.

One final note: ***TechNotes*** is planning to accept advertising starting with the September 1987 issue. For more information, call 213/538-7517.

TechNotes

Publisher
Ashton-Tate Support Services
Jay Hanson, Director

Editor-in-Chief
Brenda Johnson-Grau

Editors
Kenneth N. Getz
Karen Robinson Artan
Christopher White

Technical Staff
Larry Abare, Richard Augelli, Stanley Ballenger, Rodney Barbati, Richard Bunter, Mike Carlisle, Fred Crownover, Michael Fiore, Rick Fillman, Richard Goepel, Peter Gold, Mina Gussin, Doug Haman, Scott Haynes, Mike Hedblom, Clint Helton, Errol Jackson, Gail Kulikowsky, Steve Kurasch, Charles Litzell, Chris Lo Vasco, Jeffrey McCrimon, Jerry Mendes, Vincent Mendillo, Elizabeth Miranda, Jeffrey Moore, Keith Mund, Joseph Perret, Jaspal Rattan, John Regen, Deborah Ridolfo, Daryl Seybold, Cathy Shikatani, Steve Silverwood, Joseph Stolz, Naoma Stow, Philip Talsky, Mitra Toossi, Hiram Vega, Ralph Wakefield, Mark Ware, Quinn Wildman, Roger Wegehoft

Contributing Editors
Paul Bukauskas (International), Liz Burke (MultiMate), Steve Silverwood (On-Line Services)

Administrative Assistant
Kathy Smith

Production Staff
Gary Ashcavai, D. A. Gray Bare, Alecia C. Bernard, Tim Dunne, Darrell Evers, Renée Maynard, Patty Pesek, Ron Powell, Ann O. Taono

Product Development
Forefront Development Center: Marilyn McMahon, Greg Stikeleather
Glendale Devlopment Center: Debby Moody

DISCLAIMER NOTICE

The information contained in this publication is designed to help you develop and improve your programming skills and to assist you in using Ashton-Tate products. Ashton-Tate assumes no responsibility whatsoever for the uses made of any of the software code in this issue, whether modified or unmodified.

If you discover what you believe is an anomaly (as defined under **Anomalies**) not presented in this issue or others, please send a detailed description of the possible anomaly to:

Ashton-Tate Software Support Center
20101 Hamilton Ave.
Torrance, CA 90502-1319

If the reported anomaly can be reproduced by our technicians, it may be added to a future issue of ***TechNotes***.

The policies that are described in ***TechNotes*** apply only to the United States (and possessions), Canada, and Mexico. Residents of other foreign countries should contact a local Ashton-Tate representative in their country for policy information and technical support.

Ashton-Tate and the authors will not be liable for special, incidental, consequential, indirect or other similar damages, even if we have been advised of the possibility of such damages. This means we are not responsible for damages or costs incurred as a result of loss of time, loss of data, loss of profits or revenues, or loss of use of the software. In addition, we are not responsible for damages or costs incurred in connection with obtaining substitute software, claims by others, inconvenience or similar costs.

No warranties, either expressed or implied, are made regarding the contents of this publication, its merchantability or fitness for any particular purpose, all of which are hereby expressly disclaimed. Note, however, that some states do not allow the exclusion or limitation of warranties or of direct, incidental or consequential damages, so these warranty limitations may not apply to you.

In no event will our liability for any damages to you or any other person ever exceed the price paid for your subscription to ***TechNotes***, regardless of any form of the claim.

Additional statements by agents, employees, distributors and dealers of Ashton-Tate do not constitute warranties by Ashton-Tate and do not bind Ashton-Tate.

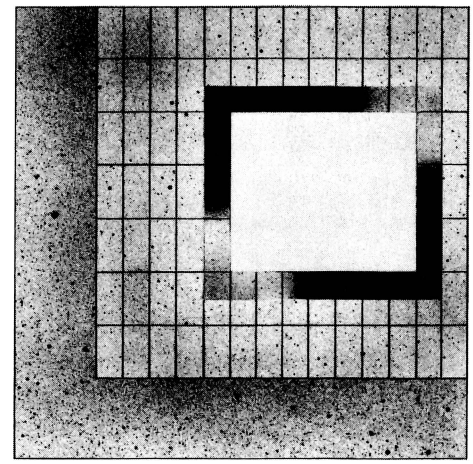
TechNotes (ISSN 0883-6493) is an Ashton-Tate Software Support Center publication containing technical reference information on Ashton-Tate and MultiMate products.

Entire contents Copyright © Ashton-Tate Corporation 1987. All rights reserved. Commercial reproduction of the material appearing in ***TechNotes*** is forbidden without prior written permission. Direct all requests for reproduction permission to Ashton-Tate Support Services, 20101 Hamilton Ave., Torrance, CA 90502-1319 ATTN: Brenda Johnson-Grau.

TechNotes is published monthly at a subscription price of $50.00 for one year. Additional rates: Canada and Mexico, $65.00; other countries, $75.00. Monthly issues are available in back issues for $10.00 each (1985 issues) and $7.00 each (1986 and 1987 issues). ***TechNotes*** is published by Ashton-Tate, 20101 Hamilton Ave., Torrance, CA 90502-1319. POSTMASTER: Send subscription inquiries and address changes to ***TechNotes*** Subscription Service, P.O. Box 3729, Escondido, CA 92025-0929.

dBASE, dBASE II, dBASE III, Ashton-Tate, and Framework are registered trademarks and RapidFile, dBASE III PLUS, Framework II RunTime, and FRED are trademarks of Ashton-Tate Corporation. Some of the trademarks of other companies mentioned in this issue appear for identification purposes only. MultiMate is a registered trademark and MultiMate Advantage, MultiMate GraphLink and the MultiMate Business Advantage Keyboard are trademarks of MultiMate International Corporation, an Ashton-Tate company. On-File is a trademark under license from Facts on File, Inc.

The Ashton-Tate Software Support Center's hours are 6:00 a.m. to 4:30 p.m., PST, Monday through Friday, except on Thursday when we close at 4:00 p.m. To reach our technicians for technical support, call (213) 329-0086. Be sure to have your software serial number ready for our operators.



Save and Restore Partial Screens

BY KENNETH N. GETZ

In the January, 1987, issue of ***TechNotes***, we published "Save and Restore Screens to File." Since then, many readers have written or called to make suggestions for improvements to the program. The program in this article, Newsave.ASM, is presented in response to these suggestions. The original intent of the January article was to allow you to create any sort of screen (a help screen or a data-entry screen, for example), save it to a disk file, and restore that screen during program execution. Many of you also wanted to save and restore partial screens to disk files. This new program can save and restore entire *and* partial screens. We have tested Newsave on IBM monochrome, IBM CGA, IBM EGA, Hercules, and Compaq display adaptors.

Newsave.BIN includes these features:

1. It can save any rectangular region of the screen to a disk file—including the entire screen. Both Newsave and the example dBASE programs support the EGA in 43-line mode, testing for the existence of the EGA card and adjusting if the screen is in 43-line mode. (Lines.BIN is included to let your dBASE programs sense whether there is an EGA present, and, if so, how many lines are displaying on the screen.)

2. It handles both the Read and Write operations. (Since dBASE III PLUS will only allow sixteen LOADed files at a time and many large applications need access to as many .BIN files as possible, merging the Read and Write operations into one file leaves more LOADable files available for other purposes.)

3. The Write option writes out the disk file with a "signature" in its header. This header serves two functions. If a user tries to TYPE the screen file from DOS, only the words "SCREEN FILE" appear on the screen, and the Read option checks to make sure the signature exists in the saved file. If it does not, the program sends an error code back to dBASE III PLUS and no screen is loaded. This prevents undesired loading of incorrect files during program execution. An attempt to TYPE Newsave.BIN from DOS results in the display of a copyright statement, not a long string of graphics characters.

Setup

To set up Newsave.BIN and Lines.BIN, create Newsave.ASM and Lines.ASM using a standard ASCII text editor (one that produces no extraneous characters of its own). Comments (anything following a semicolon in the program listing) are for human use only and will be disregarded by the assembler; so, you needn't type them. All other punctuation and text is vital to the correct assembly of the .BIN files.

Once you have created the .ASM files, run the following series of programs to convert them into executable files. Though this code was written and tested exclusively with Microsoft's MASM 4.0, it should work correctly with version 2.0 or higher of IBM or Microsoft's MASM. (No versions lower than 2.0 of MASM and no versions of ASM.EXE will work because several changes have been made to the assembler). You'll also need LINK.EXE and EXE2BIN.EXE (both supplied with DOS). Be sure that all of these programs are in the current directory or in your DOS path.

1. From the DOS prompt, type

   ```
   MASM Newsave;
   ```

   to create an object file (Newsave.OBJ).

2. Then type

   ```
   LINK Newsave;
   ```

   to create Newsave.EXE. Ignore the "Warning: no stack segment" message.

3. Type

   ```
   EXE2BIN Newsave
   ```

   to convert the program into a format dBASE III PLUS can LOAD.

4. Delete extraneous files

   ```
   ERASE Newsave.EXE
   ERASE Newsave.OBJ
   ```

5. Repeat these operations for Lines.ASM.

Passing Parameters to Newsave

Newsave requires you to place several parameters in a character string before you CALL it. To pass these parameters, you should create a memory variable, since a CALLed program can only indicate an error if it has an address through which to pass it.

You send row and column locations to Newsave as a single byte each, using the CHR() function. dBASE III PLUS can neither send nor receive CHR(0); so, all row and column values must be offset by 1. Row numbers in dBASE III PLUS go from 0 to 24, and columns go from 0 to 79. For the sake of this program, you should treat all rows as if numbered from 1 to 25, and all columns as if numbered from 1 to 80.

When writing to a disk file, Newsave needs a "W" to indicate a write, plus the four coordinates of the screen rectangle (upper left row, upper left column, lower right row, lower right column) all offset by one, as well as the filename to write to. For example, to write the portion of a screen from row 4, column 9 to row 11, column 19 to the file Scrnfile.SCR:

```
str = "W" + CHR(5) + CHR(10) + CHR(12) + CHR(20) + "Scrnfile.SCR"
CALL Newsave WITH str
```

When reading, Newsave needs an "R" to indicate a read, the coordinates of the upper left corner to write to, and the filename to read from. For example, to read the Scrnfile.SCR onto the screen at row 5, column 20:

```
str = "R" + CHR(6) + CHR(21) + "Scrnfile.SCR"
CALL Newsave WITH str
```

The memory variable holding the parameter string returns with ASCII character 175 (> >) as its first character if any error occurs. You can test for this with this error trap:

```
IF LEFT(str, 1) = CHR(175)
* ---An error has occurred
ENDIF
```

You should construct a memory variable to pass the parameters, given that dBASE III PLUS can only indicate an error if it has an address at which to put it. CALLing Newsave WITH a literal string, instead of a memory variable, doesn't pass dBASE III PLUS an address. An error could occur if you try and read from or write to a nonexistent drive, or if you tried to read from a file that can't be found or isn't in the correct format. By testing for an error condition after CALLing Newsave, you can take appropriate action in your programs.

No trapping is performed on coordinate ranges during the write operation; you must be careful that the maximum values of the row and column are correct. When reading a file, Newsave correctly adjusts a screen image that would be placed off the screen if it were restored at the requested location. For example, if you request an image that is 10 rows by 20 columns to be read onto the screen at row 20, column 70, it will actually be displayed at row 15,column 60. Newsave moves the retrieved image as little as possible to make it fit on the screen. See Figure 1.

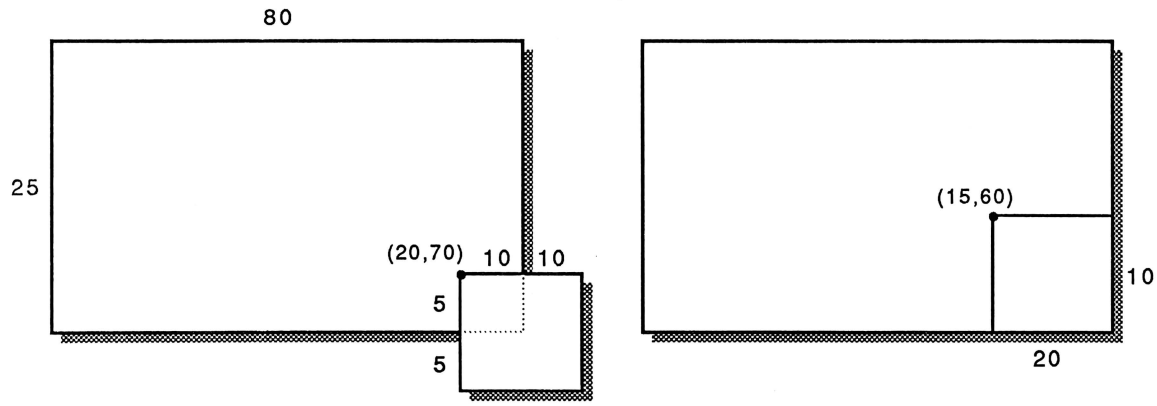


Figure 1.

An Explanation of the Newsave Code

Newsave begins by copying the entire parameter string into a local buffer. From there, it parses all the parameters into their various data locations and then determines the screen mode so that reading from and writing to screen memory is addressed correctly.

If writing a screen to disk, Newsave treats the Color Graphics Adaptor (CGA) differently from a Monochrome Display Adaptor. It writes the entire screen to an internal buffer first and thus can avoid some on-screen "snow" by sensing the retrace signal and by reading directly from screen memory only when it's safe. After the screen is read in, the correct portion of the buffer is written out to a disk file. For a monochrome screen, Newsave accesses screen memory unbuffered and writes the requested portion out to the disk.

When reading information from the disk, Newsave buffers the data no matter what the screen mode. The entire screen image can therefore be read in before anything is written to the screen. Direct writes to CGA memory are checked for the status of the retrace signal to avoid as much snow as possible.

```
; Program ...: Newsave.ASM
; Author ....: Kenneth N. Getz
; Date ......: June 1, 1987
; Version ...: dBASE III PLUS or the Developer's Release
;
; Note(s):
;
; NewSave.BIN is a routine to be LOADed and CALLed from within
; dBASE III PLUS, allowing the user to save and restore screens to disk files.
; Since this is not very useful from the DOS level, no instructions for
; conditional assembly are included.  In the following examples, row1 and col1
; refer to the screen coordinates of the upper left corner, and row2 and col2
; refer to the coordinates of the lower right corner.  Since dBASE III PLUS  cannot
; send a CHR(0) on CALL...WITH command line, all coordinates are offset by 1.
; Therefore, the upper left corner of the screen is 1, 1 and the lower right
; corner is 25, 80.
;
```

```
;   USAGE:  (to save a screen)
;   STORE "W" + CHR(row1) + CHR(col1) + CHR(row2) + CHR(col2) + filename TO x
;   CALL Newsave WITH x
;
;   (to restore a screen)
;   STORE "R" + CHR(row1) + CHR(col1) + filename TO x
;   CALL Newsave WITH x
;
;   If a file error occurs, CHR(175) is placed in the first
;   character of the memory variable passed.
;
;*****************************************************************************
MROWS   EQU     43              ; Maximum number of screen rows.
MCOLS   EQU     80              ; Maximum number of screen cols.
DOS     EQU     21h             ; Call to DOS.
CREATE  EQU     3Ch             ; Create a file.
OPEN    EQU     3Dh             ; Open a file.
CLOSE   EQU     3Eh             ; Close a file.
READ    EQU     3Fh             ; Read from a file.
WRITE   EQU     40h             ; Write to a file.
LENGTH  EQU     MROWS*MCOLS*2   ; Length in bytes of screen memory.
MONOSCR EQU     0B000h          ; Segment address of monochrome screen memory.
CLRSCR  EQU     0B800h          ; Segment address of color screen memory.
ERRCHAR EQU     175             ; Character to indicate error.
LINELEN EQU     160             ; Length of a single line in bytes.
FILERR  EQU     0FFh            ; Indication of file error.
;-----------------------------------------------------------------------------
SAVE    MACRO   R1,R2,R3,R4,R5,R6,R7,R8
; This macro takes a parameter list and pushes each parameter register, in the
; order given, onto the stack.
;
;       USAGE:  SAVE AX,BX,CX...
;
        IRP     X,<R1,R2,R3,R4,R5,R6,R7,R8>
                IFNB    <X>
                        PUSH X
                ENDIF
        ENDM
        ENDM
;-----------------------------------------------------------------------------
RESTORE MACRO   R1,R2,R3,R4,R5,R6,R7,R8
; This macro takes a parameter list and pops each parameter register, in the
; opposite of the order given, from the stack.
;
;       USAGE:  RESTORE AX,BX,CX...
;
        IRP     X,<R8,R7,R6,R5,R4,R3,R2,R1>
                IFNB    <X>
                        POP X
                ENDIF
        ENDM
        ENDM
;-----------------------------------------------------------------------------
GETFILE MACRO   MODE
LOCAL   L1,L2
; This macro opens a file, putting the file handle into the variable HANDLE.
; If an error occurs, 0FFh is put in HANDLE.  The ASCIIZ string for the file
```

```
; name is passed in by dBASE III PLUS  in the DS:BX pair.
;
;       USAGE:  GETFILE CREATE
;       USAGE:  GETFILE READ
;
        XOR     CX,CX                   ; Set off all attribute bits.
        MOV     DX,OFFSET CS:FILENM     ; INT 21h needs address in DS:DX
        MOV     CS:HANDLE,FILERR        ; Get set for file error.
        XOR     AX,AX                   ; Clear out AX.
        MOV     AH,MODE                 ; Get the correct mode in AH.
        INT     DOS                     ; Call DOS.
        JC      L1                      ; Check for carry flag set (ERROR!)
        MOV     CS:HANDLE,AX            ; If OK, move handle to storage.
L1:     NOP
        ENDM
;-------------------------------------------------------------------------------
GETMODE MACRO
; Returns the screen mode in the variable SCREEN.  SCREEN is, by default,
; 0B800H, the color screen area.  If GETMODE finds that we're using a mono
; screen, SCREEN is set to 0B000H.
;
;       USAGE:  GETMODE
;
        SAVE    AX,BX
        MOV     AH,0Fh                  ; Get set for mode check.
        MOV     BH,0                    ; Screen page into BH.
        INT     10h                     ; Call the BIOS for check.
        CMP     AL,7                    ; See if it's a monochrome screen.
        JNE     DONE                    ; If not, quit.
        MOV     CS:SCREEN,MONOSCR       ; If so, move address to SCREEN.
DONE:   RESTORE AX,BX
        ENDM

;*******************************************************************************
CODE    SEGMENT BYTE PUBLIC     'CODE'
;-------------------------------------------------------------------------------
NEWSAVE PROC    FAR
        ASSUME  CS:code
START:  JMP     ENTRY
;= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
        DB      13,'NewSave.BIN',10,13
        DB      'Written by Kenneth N. Getz.',10,13
        DB      'Copyright 1987, Ashton-Tate',10,13
        DB      26
NUMROWS DB      25                      ; Storage for actual number of rows.
REALLEN DW      80*25                   ; Storage for actual WORDS per screen.
RW      DB      0
ROW1    DB      0
COL1    DB      0
ROWS    DB      0
COLS    DB      0
FILENM  DB      65 DUP(0)               ; Storage for file name.
PARSTR  DB      70 DUP(0)               ; Buffer for param string from dBASE III PLUS.
SCREEN  DW      CLRSCR                  ; Assume we have color screen for now.
HANDLE  DW      0                       ; Storage for file handle.
SIG     DB      13                      ; Beginning of signature.
TEST    DB      'Screen File'           ; Signature to compare header to.
```

```
        DB      10,26                   ; End of signature.
HEADER  DB      16 DUP(0)               ; Storage for header read from file.
OP      DB      ?                       ; Storage for Read/Write flag.
BUFFER  DB      LENGTH DUP(0)           ; Screen save area if we have color.
;= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
ENTRY:  SAVE    AX,BX,CX,DX,SI,DI       ; Save the environment.
        PUSH    DS                      ; Save dBASE III PLUS's DS.
        CALL    PARAMS                  ; Get the passed parameters.
        GETMODE                         ; Find the current screen mode.
        CALL    LINES
        PUSH    CS
        POP     DS                      ; Make DS = CS.
        CMP     RW,'R'                  ; See if we're reading or writing.
        JE      READIT                  ; If reading, go to that section.
WRITEIT:                                ; Otherwise, we're writing.
        GETFILE CREATE                  ; Create new file with name passed in.
        CMP     HANDLE,FILERR           ; Signals that error occurred.
        JNE     WRTOK                   ; If no error, continue.
        POP     DS                      ; Restore dBASE III PLUS's DS.
        JMP     ERROR                   ; Then quit.
WRTOK:  CALL    PUTHEAD                 ; Write the header to the file.
        CMP     SCREEN,MONOSCR          ; See if GetMode found a Mono screen.
        JNE     COLOR                   ; If not, go to color routines.
        MOV     DS,SCREEN               ; Otherwise, set up for screen save.
        XOR     DX,DX                   ; Clear out DX.
        JMP     CPYOUT                  ; Do the save and quit.
COLOR:  MOV     DX,OFFSET BUFFER        ; Set up for save to buffer.
        CALL    FILLBUF                 ; Move screen memory to buffer area.
CPYOUT: MOV     OP,WRITE                ; Set up for write.
        CALL    MOVMEM                  ; Write info out to file.
        JMP     CLOSEOUT                ; And quit.
READIT:
        GETFILE OPEN                    ; Open file.
        CMP     HANDLE,FILERR           ; Check for file error.
        JNE     RDOK                    ; If no error, continue.
        POP     DS                      ; Reset dBASE III PLUS's DS.
        JMP     ERROR                   ; Then finish up.
RDOK:   CLC                             ; Clear the carry flag for next test.
        CALL    VERIFY                  ; Verify file, set CF if not OK.
        JNC     NOERR                   ; File was a NEWSAVE file, so keep on.
        POP     DS                      ; Otherwise, reset DS.
        JMP     ERROR                   ; And get out.
NOERR:  MOV     DX,OFFSET BUFFER        ; Move offset of buffer into DX.
        MOV     OP,READ                 ; Set up for read.
        CALL    MOVMEM                  ; Read info to buffer.
        CALL    RDBUFF                  ; Read from buffer to screen.
CLOSEOUT:
        POP     DS                      ; Restore dBASE III PLUS's DS.
        JMP     FINISH                  ; Skip error action.
ERROR:  MOV     BYTE PTR [BX],ERRCHAR   ; Indicate error.
        JMP     OUT                     ; And quit.
FINISH: MOV     BX,CS:HANDLE            ; Get the file handle.
        MOV     AH,CLOSE                ; Request file close operation.
        INT     DOS                     ; Execute the file close.
OUT:    RESTORE AX,BX,CX,DX,SI,DI       ; Restore the environment.
        RET                             ; And return to dBASE III PLUS.
NEWSAVE ENDP
```

```
;-------------------------------------------------------------------------
CLRREAD PROC    NEAR
;
; This procedure writes a memory buffer to the screen memory after
; reading from a disk file.  This procedure's timing is very tight, and it was
; written to minimize screen "snow."  It works correctly on B&W, Color, and
; EGA graphics cards that have been tested.
;
        SAVE    AX,BX,CX,DX,BP,DS,SI,DI
        MOV     DX,3DAh                 ; Point DX to CGA status port.
        MOV     CL,COLS                 ; Get number of columns to move.
R1:     LODSW                           ; Get a video word.
        MOV     BP,AX                   ; Store it in BP.
        MOV     AH,9                    ; Get horiz and vert retrace mask.
R2:     IN      AL,DX                   ; Get 6845 status.
        RCR     AL,1                    ; Check horizontal retrace.
        JB      R2                      ; Loop if in horiz retrace.
        CLI                             ; No ints during critical section.
R3:     IN      AL,DX                   ; Get 6845 status.
        AND     AL,AH                   ; Check for both kinds of retrace.
        JNZ     R3                      ; Loop if not equal zero.
        MOV     AX,BP                   ; Get the video word.
        STOSW                           ; Store the video word.
        STI                             ; Allow interrupts.
        LOOP    R1                      ; Go do next word.
        RESTORE AX,BX,CX,DX,BP,DS,SI,DI
        RET
CLRREAD ENDP
;-------------------------------------------------------------------------
FILLBUF PROC    NEAR
;
; This procedure writes the screen memory to a memory buffer for later
; writing to a disk file.  This procedure's timing is very tight, and it was
; written to minimize screen "snow."  It works correctly on B&W, Color, and
; EGA graphics cards that have been tested.
;
        SAVE    AX,BX,CX,DX,BP,DS
        PUSH    CS
        POP     ES                      ; Make CS and ES point to same segment.
        MOV     DI,OFFSET BUFFER        ; Point DI to beginning of buffer.
        MOV     DS,SCREEN               ; Point DS to screen segment.
        XOR     SI,SI                   ; Point SI to beginning of screen.
        MOV     DX,3DAh                 ; Point DX to CGA status port.
        MOV     CX,CS:REALLEN           ; Move length value into CX.
        CLD                             ; Set string direction to forward.
W1:     IN      AL,DX                   ; Get 6845 status.
        RCR     AL,1                    ; Check horizontal retrace.
        JB      W1                      ; Loop if in horizontal retrace.
        CLI                             ; No interrupt during critical section.
W2:     IN      AL,DX                   ; Get 6845 status.
        RCR     AL,1                    ; Check for horizontal retrace.
        JNB     W2                      ; Loop if not in retrace.
        LODSW                           ; Load the video word.
        STI                             ; Allow interrupts.
        STOSW                           ; Store the video word.
        LOOP    W1                      ; Go to next word.
        RESTORE AX,BX,CX,DX,BP,DS
```

```
        RET
FILLBUF ENDP
;---------------------------------------------------------------------------
LINES     PROC NEAR
;
; This procedure finds out the current number of lines per screen and
; uses this to correct all values for lines per screen different than 25.
; Specifically written to cover the EGA 43-line mode.
;
     SAVE    AX,BX,CX,DX,ES
     SUB     DL,DL                  ; Zero out DL.
     SUB     BH,BH                  ; Zero out BH.
     MOV     AX,1130H               ; Set up for interrupt call.
     INT     10H                    ; Call INT 10h.
     OR      DL,DL                  ; Check to see if DL is 0.
     JZ      LINEOUT                ; If so, we don't have EGA.
     INC     DL                     ; Otherwise, bias number to 1.
     MOV     NUMROWS,DL             ; Store the number in the data area.
     MOV     AX,MCOLS               ; Get set to multiply.
     MUL     DL                     ; Figure total number screen positions.
     MOV     REALLEN,AX             ; Store this in the data area.
LINEOUT:
     RESTORE AX,BX,CX,DX,ES
     RET
LINES        ENDP
;---------------------------------------------------------------------------
MOVMEM  PROC    NEAR
; This routine calls a DOS interrupt to write a specific area of memory out
; to a disk file (specified by the file handle).  It expects DS:DX to be
; pointing to the beginning of memory to be written.
;
        SAVE    AX,CX                  ; Save environment.
        MOV     AL,ROW1                ; Get the starting row.
        MOV     CL,LINELEN             ; Get length of each line.
        MUL     CL                     ; Multiply for offset into buffer.
        ADD     DX,AX                  ; Add onto starting position.
        MOV     CL,COL1                ; Get starting column.
        SHL     CL,1                   ; Multiply by two to account for attr.
        ADD     DX,CX                  ; Add onto starting position.
        XOR     CH,CH                  ; Clear out high byte.
        MOV     CL,ROWS                ; Set up number of rows to move.
MOVELOOP:
        PUSH    CX                     ; Save row number.
        MOV     CL,COLS                ; Get the number of cols to be read.
        SHL     CL,1                   ; Multiply by two to account for attr.
        MOV     BX,HANDLE              ; Get the file handle.
        MOV     AH,OP                  ; Get set to do the read.
        INT     DOS                    ; Do the read.
        ADD     DX,LINELEN             ; Move on to next line.
        POP     CX                     ; Restore the line number.
        LOOP    MOVELOOP               ; And go back for more.
        RESTORE AX,CX                  ; Restore the environment.
        RET
MOVMEM  ENDP
;---------------------------------------------------------------------------
PARAMS  PROC    NEAR
;
```

```
; This procedure fills the data area with arguments passed on the command
; line from dBASE III PLUS.  This routine is very dependent on the order of the
; variables in the data area.
;
        SAVE    AX,CX,DX,DS             ; Save the environment.
        MOV     CL,BYTE PTR [BX-1]      ; Get dBASE III PLUS's param string length.
        XOR     CH,CH                   ; Clear the high byte.
        DEC     CX                      ; Don't want the last NULL byte.
        MOV     SI,BX                   ; Point SI to the string.
        MOV     DI,OFFSET PARSTR        ; Point DI to the buffer area.
        REP     MOVSB                   ; Copy it all in to the code area.
        XOR     AL,AL                   ; Clear out AL.
        STOSB                           ; Append NULL byte at end of string.
        PUSH    CS                      ; Make DS = CS.
        POP     DS
        LEA     SI,PARSTR               ; Point SI to ParStr.
        LEA     DI,RW                   ; Point DI to RW (first parameter).
        LODSB                           ; Get RW (Read/Write).
        AND     AL,0DFH                 ; Convert to UPPER case.
        STOSB                           ; Store it to RW.
        LODSB                           ; Get ROW1.
        DEC     AL                      ; Decrement it (bias of 1).
        STOSB                           ; Store it to ROW1.
        LODSB                           ; Get COL1.
        DEC     AL                      ; Decrement it (bias of 1).
        STOSB                           ; Store it to COL1.
        CMP     RW,'R'                  ; See if we're Reading or Writing.
        JE      GETFILENM               ; If reading, jump to get filename.
        LODSB                           ; Get ROW2 to calc number of rows.
        SUB     AL,ROW1                 ; Subtract the two.
        STOSB                           ; And store it to ROWS.
        LODSB                           ; Get COL2 to calc number of cols.
        SUB     AL,COL1                 ; Subtract COL1.
        STOSB                           ; And store it in COLS.
GETFILENM:
        MOV     CX,64                   ; Maximum path length is 64.
        LEA     DI,FILENM               ; Point DI to the filename.
FILELOOP:
        LODSB                           ; Get a character.
        CMP     AL,0                    ; See if it's the end.
        JZ      FILEDONE                ; If so, jump out.
        STOSB                           ; Otherwise, store the byte.
        LOOP    FILELOOP                ; And go back for more.
FILEDONE:
        STOSB                           ; Store another null for termination.
        RESTORE AX,CX,DX,DS             ; Restore the environment.
        RET
PARAMS  ENDP
;---------------------------------------------------------------------------
PUTHEAD PROC    NEAR
;
; This procedure writes out the header to the saved screen file.  The
; header contains a byte for the number of rows, a byte for the number
; of columns, and the string "Screen File" which is both displayed if the
; user TYPEs the file out, and is verified at loading time to make sure
; the file was saved by Newsave.
;
```

```
        SAVE    AX,BX,CX,DX
        MOV     BX,HANDLE               ; Get the file handle.
        MOV     DX,OFFSET ROWS          ; Get the address of info to write.
        MOV     CX,2                    ; Get the number of bytes to write.
        MOV     AH,WRITE                ; Request write operation.
        INT     DOS                     ; Execute the Write.
        MOV     DX,OFFSET SIG           ; Get the address of the signature.
        MOV     CX,14                   ; Get the number of bytes to write.
        MOV     AH,WRITE                ; Request write operation.
        INT     DOS                     ; Execute the write.
        RESTORE AX,BX,CX,DX
        RET
PUTHEAD ENDP
;-------------------------------------------------------------------------
RDBUFF  PROC    NEAR
;
; This procedure reads the information from the buffer and places it at the
; correct location on the screen.  While the two loops look redundant, length
; of code was balanced against speed of execution -- i.e., the branch based
; on type of display was made outside the loop (thereby executed only once)
; instead of inside the loop (shortening the code but executing a branch on
; each iteration of the loop).
;
        SAVE    AX,CX,ES                ; Save the environment.
        MOV     AL,ROW1                 ; Get the starting row.
        MOV     CL,LINELEN              ; Get the length of each line.
        MUL     CL                      ; Multiply for offset into buffer.
        LEA     SI,BUFFER               ; Point SI to beginning of buffer.
        MOV     ES,SCREEN               ; Point ES to screen memory.
        XOR     DI,DI                   ; Point DI to beginning of screen.
        ADD     SI,AX                   ; Add to SI (source) row offset.
        ADD     DI,AX                   ; Add to DI (dest) row offset.
        MOV     CL,COL1                 ; Get column offset.
        SHL     CL,1                    ; Multiply by two for char and attr.
        ADD     SI,CX                   ; Add this to source pointer.
        ADD     DI,CX                   ; Add this to dest pointer.
        XOR     CH,CH                   ; Zero out CH.
        MOV     CL,ROWS                 ; Get the number of rows to show.
        CLD                             ; Clear the direction flag.
        CMP     SCREEN,MONOSCR          ; See if GetMode found a mono screen.
        JNE     CLRLOOP                 ; If not, go to color branch.
MONOLOOP:
        PUSH    CX                      ; Save the row counter.
        SAVE    SI,DI                   ; Preserve offset pointers.
        MOV     CL,COLS                 ; Get the number of columns.
        REP     MOVSW                   ; Move the correct number of words.
        RESTORE SI,DI                   ; Restore offset pointers.
        ADD     DI,LINELEN              ; Move on to next line.
        ADD     SI,LINELEN
        POP     CX                      ; Restore row counter.
        LOOP    MONOLOOP                ; Go back for more rows.
        JMP     CLROUT                  ; And close out.
CLRLOOP:
        CALL    CLRREAD                 ; Read a single row.
        ADD     DI,LINELEN              ; Move on to next line.
        ADD     SI,LINELEN
        LOOP    CLRLOOP                 ; And go back for more.
```

```
CLROUT: RESTORE AX,CX,ES                ; Restore environment.
        RET
RDBUFF  ENDP
;---------------------------------------------------------------------------
VERIFY  PROC    NEAR
;
; This procedure is called to verify that the file being accessed is a file
; that was created using the Newsave.BIN program.  If not, all sorts of fun
; things could happen.  It first reads the row and column counters from the
; first two bytes of the file, and then checks to make sure that the file
; includes the signature "Screen File" before loading it.  This sets the
; Carry Flag if an error occurs.
;
        SAVE    AX,BX,CX,DX,SI,DI       ; Save the environment.
        CLD                             ; Clear DF so increments left to right.
        MOV     BX,HANDLE               ; Get the file handle.
        MOV     CX,16                   ; We're going to read 16 bytes.
        LEA     DX,HEADER               ; Set the output buffer.
        MOV     AH,READ                 ; Get set to read.
        INT     DOS                     ; Read from the file.
        LEA     DI,TEST                 ; Point DI to the correct string.
        LEA     SI,HEADER               ; Point SI to the header from the file.
        LODSB                           ; Get one byte.
        MOV     ROWS,AL                 ; Move it to ROWS.
        LODSB                           ; Get one byte.
        MOV     COLS,AL                 ; Move it to COLS.
        MOV     AL,NUMROWS              ; The max number of rows.
        SUB     AL,ROWS                 ; Subtract off number of rows in file.
        SUB     AL,ROW1                 ; Subtract off starting row.
        CMP     AL,0                    ; See if we're under 0.
        JGE     ROWS_OK                 ; If not, we're OK.
        ADD     AL,ROW1                 ; Otherwise, add back starting row,
        MOV     ROW1,AL                 ;   which will now be less than it was.
ROWS_OK:
        MOV     AL,MCOLS                ; The max number of cols.
        SUB     AL,COLS                 ; Subtract off number of cols in file.
        SUB     AL,COL1                 ; Subtract off starting col.
        CMP     AL,0                    ; See if we're under 0.
        JGE     COLS_OK                 ; If not, we're OK.
        ADD     AL,COL1                 ; Otherwise, add back starting col,
        MOV     COL1,AL                 ;   which will now be less than it was.
COLS_OK:
        INC     SI                      ; Skip over EOF byte.
        MOV     CX,11                   ; Set to read 11 bytes.
        REPE    CMPSB                   ; Compare all the bytes.
        JE      MATCH                   ; If through all of them, we're OK.
        STC                             ; Otherwise, set the Carry Flag.
MATCH:  RESTORE AX,BX,CX,DX,SI,DI       ; Restore the environment.
        RET
VERIFY  ENDP
;---------------------------------------------------------------------------
CODE    ENDS
        END     START
```

Passing Parameters to Lines

Lines just needs a location to return the current number of lines per screen if there is an EGA card, or a CHR(255) if there is not. To use it, just declare a variable that contains a single space, and then CALL Lines WITH that variable.

A typical use might look like:

```
numlines = SPACE(1)
CALL Lines WITH numlines
IF numlines = CHR(255)
     ? "There is no EGA"
ELSE
     ? "There are " + STR(ASC(numlines),2) + " lines displayed."
ENDIF
```

```
; Program ...: Lines.ASM
; Author ....: Kenneth N. Getz
; Date ......: June 1, 1987
;
; Note(s):
;
; This .BIN file informs you if you are using an EGA card,
; and if so, how many lines currently on the screen.  The routine must
; be LOADed first, and then CALLed with the following syntax:
;
;    LOAD Lines
;    numlines = SPACE(1)
;    CALL Lines WITH numlines
;
; If numlines = CHR(255), then there is no EGA installed.  Otherwise, numlines
; contains a character corresponding to the current number of lines on screen.
; If there are 25 lines, numlines = CHR(25), and if there are 43 lines,
; numlines = CHR(43).
;
;
CODE    SEGMENT BYTE    PUBLIC  'CODE'
LINES   PROC    FAR
        ASSUME  CS:CODE
        PUSH    BX                      ; Preserve registers.
        PUSH    ES                      ; Since ES is damaged, preserve it
too.
        SUB     DL,DL                   ; Clear out DL.
        SUB     BH,BH                   ; Clear out BH.
        MOV     AX,1130H                ; Set up for interrupt call.
        INT     10H                     ; Call the EGA BIOS.
        OR      DL,DL                   ; See if we got a zero.
        JZ      NOEGA                   ; IF so, we're done.
        INC     DL                      ; Otherwise, bias the number to 1.
        JMP     FINISH                  ; And skip out.
NOEGA:  MOV     DL,255                  ; If no EGA, send back CHR(255).
FINISH: POP     ES                      ; Restore registers.
        POP     BX
        MOV     BYTE PTR [BX], DL       ; Send back the required value.
        RET                             ; Back to dBASE.
LINES   ENDP
```

```
CODE    ENDS
        END
```

An Application Example

MakeScr.PRG and Disptest.PRG are two dBASE III PLUS programs written to test Newsave.BIN. MakeScr creates a rectangular display and saves it to a disk file named "Help.SCR". Disptest displays the file Help.SCR over a screen full of numbers. The program alternately restores the small portion of the screen and the full screen, allowing you to move the small region to each of the four corners of the screen by pressing **Home**, **End**, **PgUp** or **PgDn**. You can use these programs as prototypes for your own save-and-restore screen routines.

```
* Program ...: MakeScr.PRG
* Author ....: Kenneth N. Getz
* Date ......: June 1, 1987
* Version ...: dBASE III PLUS or the Developer's Release
* Note(s) ...: This program generates the screen necessary for DispTest.PRG.
*              Make sure and run this program before DispTest.
*
LOAD Newsave
CLEAR
@  0,  3 CLEAR TO  9,41
@  0,  3 TO  9, 41 DOUBLE
@  2,  4 TO  2, 40
@  7,  4 TO  7, 40
@  1,  5 SAY "Sample HELP Screen"
@  3,  5 SAY "You can move this box to all"
@  4,  5 SAY "four corners of the screen by"
@  5,  5 SAY "pressing Home, PgUp, End, or PgDn"
@  8,  9 SAY "Press Return or Esc to Exit."
*
* ---Note that the values given on the next line are all one more
* ---than the values given in the box-drawing statement above.
STORE "W" + CHR(1) + CHR(4) + CHR(10) + CHR(42) + "Help.SCR" TO x
CALL Newsave WITH x
CLEAR
IF LEFT(x, 1) = CHR(175)
   ? "  There was a problem creating your screen file."
   ? "  Please check the parameters and try again."
ENDIF
RETURN
* EOP MakeScr.PRG
```

```
* Program ...: Disptest.PRG
* Author ....: Kenneth N. Getz
* Date ......: June 1, 1987
* Version ...: dBASE III PLUS or Developer's Release
* Note(s) ...: Program to show NewSave to saving and restoring areas
*              of screens.  Uses a screen file (Help.SCR)
*              that must be prepared before running Disptest.
*
*              To create Help.SCR, run the program MakeScr.PRG first.
*
* ---Set up the environment.
CLEAR
SET TALK OFF
SET STATUS OFF
SET ESCAPE OFF
SET SCOREBOARD OFF
LOAD Newsave
LOAD Lines
*
* ---Initialize the Save/Restore variables.
maxrows   = 25                      && Default max rows per screen.
maxcols   = 79                      && Maximum columns per screen.
bigfile   = "Full.SCR"
smallfile = "Help.SCR"              && Help.SCR is your "help" screen.
numlines  = SPACE(1)
*
* ---See if there is an EGA, and adjust lines per screen.
CALL Lines WITH numlines
IF numlines <> CHR(255) .AND. numlines <> CHR(25)
   maxrows = ASC(numlines)
ENDIF
fullsave  = "W" + CHR(1) + CHR(1) + CHR(maxrows) + CHR(maxcols) + bigfile
fullrest  = "R" + CHR(1) + CHR(1) + bigfile
*
* ---Initialize variables dealing with screen positions.
row       = 1                       && Starting row for help screen.
col       = 1                       && Starting column for help screen.
newrow    = 1
newcol    = 1
totalrows = 9                       && Rows of the restored screen.
totalcols = 38                      && Cols in restored screen.
*
* ---Initialize key table.
homekey  = CHR(1)
pgdn     = CHR(3)
endkey   = CHR(6)
retkey   = CHR(13)
pgup     = CHR(18)
esc      = CHR(27)
highval  = CHR(255)
ok_keys  = homekey + pgdn + endkey + retkey + pgup + esc
out_keys = retkey + esc
*
* ---Draw a "background" screen for testing and save it.
i = 1
```

```
DO WHILE i < maxrows
   @ i - 1,  0 SAY "This is row " + LTRIM(STR(i))
   @ i - 1, 15 SAY "56789" + REPLICATE("0123456789", 6)
   i = i + 1
ENDDO
CALL Newsave WITH fullsave
*
* ---Put the help screen up at row - 1, col - 1.
var = "R" + CHR(row) + CHR(col) + smallfile
CALL Newsave WITH var
*
* ---Get keystrokes, moving the help screen accordingly.
DO WHILE .T.
   *
   * ---Get a keystroke.
   key = highval
   DO WHILE .NOT. key $ ok_keys
      key = UPPER(CHR(INKEY()))
   ENDDO
   *
   * ---Find correct action.
   DO CASE
   CASE key $ out_keys
    EXIT
   CASE key  = homekey
    newcol = 1
      newrow = 1
   CASE key  = pgdn
      newrow = maxrows - totalrows
      newcol = maxcols - totalcols
   CASE key  = endkey
      newrow = maxrows - totalrows
      newcol = 1
   CASE key  = pgup
      newrow = 1
      newcol = maxcols - totalcols
   ENDCASE
   *
   * ---If no change, don't do anything.
   IF row = newrow .AND. col = newcol
      LOOP
   ENDIF
   *
   * ---Assign new row and col values.
   row = newrow
   col = newcol
   var = "R" + CHR(row) + CHR(col) + smallfile
   *
   * ---Restore full screen.
   CALL Newsave WITH fullrest
   *
   * ---Display help at new position.
   CALL Newsave WITH var
ENDDO
*
* ---Finally, restore original screen.
CALL Newsave WITH fullrest
```

```
CLEAR
RELEASE MODULE Lines
RELEASE MODULE Newsave
* EOP Disp.PRG
```

References

1. Borland Int'l. *Turbo Pascal Editor Toolbox*. Scotts Valley: Borland Int'l.

2. Norton, Peter. *The Peter Norton Programmer's Guide to the IBM PC*. Bellevue, Washington: Microsoft Press. pps. 67-97.

Snaked Column Reporting

BY SOFTWARE SUPPORT TECHNICIANS

If you need to print directories, parts lists, or catalogs, you may have wondered how to create a report in which the columns wrap like a standard phone book. You probably realized quite quickly that REPORT FORM cannot wrap from one column to the next, although it can wrap within a column. You may then have viewed LABEL FORM with a hopeful eye, but found that although you can create column reports, you cannot create a column of records and then place beside it another column of records. What you need is a custom report program that performs the application for you. The following program, Snake.PRG, does exactly this.

Snaked column reporting, if you are not familiar with the term, refers to the way column contents are laid out on the page. In a standard column-oriented report such as REPORT FORM, columns are report fields where each row or group of rows is derived from a record in your database file or View. Columns are related by row. Snaked column reports, by contrast, are like newspaper columns. Row order exists only within a column and not across columns. Page columns are linked bottom to top, the last record in the first column is followed by the first record in the second column giving you the snaking effect. Figure 1 shows the relationship of columns and rows.

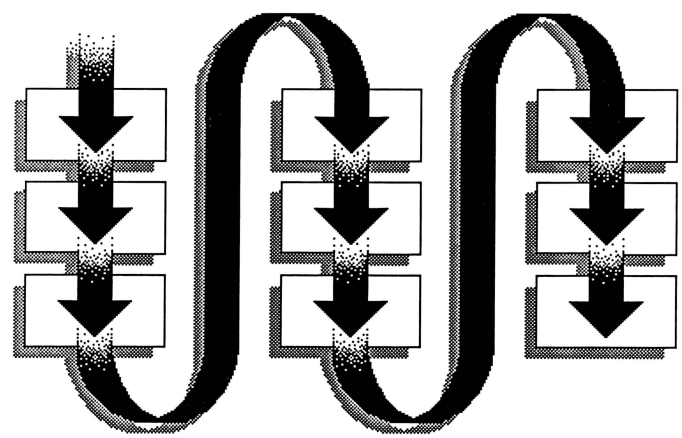


Figure 1.

Snake.PRG is designed to be as easy to use and as automatic as possible. Basically, all you do to run it is to pass parameters for the data definition (database file, index file, and filter condition), page format (header and footer), and column definition (column heading, column contents expression, and record number flag). The program figures out the rest and, based on a series of configurable memory variables listed at the top of Snake.PRG, prints the report.

Each report page has standard dimensions you can change. When the snake column report prints, each column is separated by a configurable delimiter string in addition to the beginning and end of each row. The default value of this string is the character "|", but you can make it any character or string of characters you want. The delimiter variable is not a passed parameter but is initialized in the user definition table at the top of Snake.PRG.

The column contents is an expression that outputs the display for each column. This expression must be character type and can consist of fields, memory variables, and literal strings. When Snake places the result of the contents expression in the column, it always left-justifies it within the column. If you need to print two fields within a column, one left-justified and one right-justified, your contents expression must take this into consideration.

For example, suppose you have two fields, Name and Phone. In your snaked column report, you want to print Name left-justified and Phone right-justified, separated by a series of dots like the following:

```
Jack Smith     ....... 555-1212
Marilyn Smith ....... 555-1212
```

To accomplish this, pass the following as the column contents expression parameter on the DO Snake command line:

```
                                                  Column width
TRIM(Name) + SPACE(1) + REPLICATE(".", 25 - LEN(TRIM(Name) +;
SPACE(2) + TRIM(Phone))) + SPACE(1) + Phone
```

The consideration here is that the column length must be known prior to running the report since it is used to determine the number of intermediate characters between the left-justified field and the right-justified field. Snake fortunately has a feature that accounts for this. If you want to find out the column width from the parameters you pass, create a numeric memory variable "colwidth." Then DO Snake WITH your report parameters. If "colwidth" exists, Snake calculates report column width placing it in "colwidth" and RETURNs to the calling program without printing the report. With this facility, you can DO Snake once to obtain the column width and then use the value returned as a part of the report field expression if it needs the column width as an argument. For example,

```
colwidth = 0
*
* ---Run Snake to determine the report width.
DO Snake WITH <dummy parameters>
*
* ---Assign "colwidth" to variable you can use in your report
* ---field expression and get rid of the "colwidth" so that
* ---your report will print.
newidth = colwidth
```

```
RELEASE colwidth
*
* ---Run Snake with the real parameter list using "newidth"
* ---in the report field expression.
DO Snake WITH <real parameters>
```

Snake provides some other features. These include optional page, record numbering, and column delimiter. Specifying the string "<#>" in either the header or footer expression places the current page number in the position where you placed the special symbol. Specifying 0, 1, or 2 for the record number flag determines whether an optional record number is placed next to each record's contents. Zero suppresses the automatic numbering, 1 places the actual record number beside the record contents, and 2 places a number representing the position of the record in the printed list. You can also specify a column heading that is placed left-justified above each column on every page of the report.

Setup

To set up, create the file Snake.PRG using the text editor of your choice. Then enter the program code for Snake and the procedures Reptproc, Spacing, and Getpage into Snake.PRG. If this seems a little strange, don't be alarmed. dBASE III PLUS permits a command file to SET PROCEDURE TO itself.

If you wish to change any of the fundamental behaviors of the report, you can change any of the variables in the user definition table at the top of Snake.PRG. For specific instructions as to the legal values for variables refer to the comments within the user definition table.

Usage

In order to begin the program from the dot prompt, issue the following command:

```
DO Snake WITH <database file>, [<index file>], [<filter condition>],;
              [<header>], [<footer>],;
               <column contents expression>, [<column heading>];
               <number of columns>, <record number flag>
```

As you can see from this general syntax statement, there are a number of optional parameters. If you do not have a value for a specific parameter, pass a null string ("") in its place.

When you run Snake.PRG, it sends its output to the screen. To direct output to the printer, SET PRINT ON before you run it and SET PRINT OFF after it completes. Remember that since it outputs using ?, to suppress output to the screen, SET CONSOLE OFF before you SET PRINT ON. In the same way, you direct output to the printer, you can direct it to an alternate file using SET ALTERNATE. The only consideration here is that the number of lines Snake prints per page is set for 66 (the standard print page length). If you want to change this, change the memory variables, pf_pl, pf_tm, and pf_bm in the user definition table. Moreover, if you plan to output with Snake to a number of different devices, consider adding these memory variables to the parameter list so you can pass them from the calling program.

```
* Program ...: Snake.PRG
* Author ....: Olivier Biggerstaff
* Date ......: June 1, 1987
* Note(s) ...: Parameters to be passed as:
*
*     pq_dbf    ::= database to be used, character string
*     pq_ndx    ::= index file to be used, character string
*     pq_query  ::= filter condition, character string
*     pc_header ::= page header, character string
*     pc_footer ::= page footer, character string
*     disp_exp  ::= expression to be printed, character string
*     colhead   ::= heading for each column, character string
*     pc_cols   ::= number of columns
*     pc_recnum ::= status of record number printing, integer
*                   0 for no record numbers
*                   1 for actual record number
*                   2 for ascending list
*
PARAMETERS pq_dbf, pq_ndx, pq_query, pc_header, pc_footer,;
           disp_exp, colhead, pc_cols, pc_recnum
SET TALK OFF
SET PROCEDURE TO Snake
pq_ndx = IIF("" = pq_ndx, pq_ndx, " INDEX " + pq_ndx)
USE &pq_dbf.&pq_ndx
SET FILTER TO &pq_query
GOTO TOP
*
* ------------------- User definition table ------------------------
*
* ---Define print format settings.
pc_cdelim = "|"                     && Column delimiters.
pf_fp     = 2                       && Footer position, measured from the bottom of page.
pf_hp     = 2                       && Header position, measured from the top of the page.
*
* ---The next three items should add up to equal the length of your page in lines.
pf_pl     = 60                      && Page length.
pf_tm     = 3                       && Top margin.
pf_bm     = 3                       && Bottom margin.
*
* ---The next two items should add up to equal the width of your page in characters.
pf_lm     = 0                       && Left margin.
pf_ll     = 79                      && Line length.
* ----------------------------------------------------------------
*
* ---Define internal memory variables.
headspace = 1                                 && Lines between column header and data.
newline   = CHR(13) + CHR(10)                 && New line characters.
recwidth = IIF(pc_recnum > 0,;
           LEN(LTRIM(STR(RECCOUNT()))), -1)   && Maximum width of record numbers.
pagechar = "<#>"                              && Indication for page number.
*
* ---If there is a header, it must be accounted for.
items = IIF("" <> colhead, pf_pl - headspace - 1, pf_pl)
*
DO Reptproc WITH disp_exp, pc_recnum
```

```
CLOSE PROCEDURE
SET FILTER TO
SET INDEX TO
USE
RETURN
* EOP Snake
```

```
PROCEDURE Reptproc
PARAMETERS disp_exp, pc_recnum
*
* ---Figure the column width.
pf_colwdth = INT(pf_ll - (pc_cols + 1) * LEN(pc_cdelim)) / pc_cols - (recwidth + 1)
*
* ---If the variable "colwidth" has a numeric value, pass back the column width
* ---into that variable and return.
IF TYPE("colwidth") = "N"
   colwidth = pf_colwdth
   RETURN
ENDIF
*
* ---Pad the display expression with spaces for later truncation.
disp_exp = disp_exp + " + SPACE(" + LTRIM(STR(pf_colwdth)) + ")"
page   = 1                                  && The current page.
recnum = RECNO()                            && The current record number.
dopage = .T.                                && Indicates another page to print.
DO WHILE dopage .AND. .NOT. EOF()
   *
   * ---Space down to header line, make sure that header will fit inside top
   * ---margin, print the header if so, and then space down the correct number of
   * ---lines to get to the first line of data (which may be a column header).
   IF pf_hp <= pf_tm
      DO Spacing WITH pf_hp - 1
      newheader = pc_header
      DO Getpage WITH newheader, page
      ? SPACE((pf_lm + pf_ll - LEN(newheader)) / 2) + newheader
      ?
      DO Spacing WITH pf_tm - pf_hp
   ELSE
      DO Spacing WITH pf_tm + 1
   ENDIF
   rownum = 1                               && On the first row of the page.
   itemnum = 1                              && On the first item of the page.
   *
   * ---Place the column header above each column and then space down to the
   * ---first line of data.
   IF "" <> colhead
      ?? SPACE(pf_lm)
      colnum = 1
      colhead = LEFT(colhead, pf_colwdth + recwidth + 1)
      headlen = LEN(colhead)
      DO WHILE colnum <= pc_cols
         ?? SPACE(LEN(pc_cdelim)) + colhead +;
            SPACE(recwidth + 1 + pf_colwdth - headlen)
```

```
      colnum = colnum + 1
   ENDDO
   DO Spacing WITH headspace + 1
   rownum = rownum + headspace + 1
ENDIF
*
* ---Now put all the data in for each row.
DO WHILE rownum <= pf_pl
   colnum = 1
   ?? SPACE(pf_lm) + pc_cdelim
   * ---Calculate the base number from which the rest of the numbers are found.
   number = (page - 1) * items * pc_cols + itemnum - items
   DO WHILE colnum <= pc_cols
      number = IIF(pc_recnum = 1, RECNO(), number + items)
      IF EOF()
         * ---If at end of file, just print spaces.
         ?? IIF(pc_recnum > 0, SPACE(recwidth + 1), "") +;
            SPACE(pf_colwdth) + pc_cdelim
         dopage = .F.
      ELSE
         * ---Otherwise, print the number if necessary and then the data truncated to fit
         * ---the column.
         ?? IIF(pc_recnum > 0, STR(number, recwidth) + " ", "") +;
            LEFT(&disp_exp, pf_ colwdth) + pc_cdelim
         lastrec = RECNO()
         SKIP items
      ENDIF
      colnum = colnum + 1
   ENDDO
   *
   * ---Update the counters and move to the correct record.
   rownum  = rownum + 1
   itemnum = itemnum + 1
   GOTO recnum
   SKIP
   recnum = IIF(EOF(), recnum, RECNO())
   *
   * ---Get set for the next row.
   number = number - items + 1
   ?
ENDDO
*
* ---Move to the footer line if the footer will fit in the bottom margin.
IF pf_fp <= pf_bm
   DO Spacing WITH pf_bm - pf_fp
   newfooter = pc_footer
   DO Getpage WITH newfooter, page
   ?? SPACE((pf_lm + pf_ll - LEN(newfooter)) / 2) + newfooter
   DO Spacing WITH pf_fp - 1
ELSE
   DO Spacing WITH pf_bm - 1
ENDIF
page = page + 1
GOTO lastrec
SKIP
recnum = RECNO()
```

```
   number = number - items + 1
ENDDO
RETURN
* EOP Reptproc
```

```
PROCEDURE Spacing
*
PARAMETERS maxspace
IF maxspace > 0
   ? REPLICATE(newline, maxspace - 1)
ENDIF
RETURN
* EOP Spacing
```

```
PROCEDURE Getpage
*
PARAMETERS string, pagenum
position = AT(pagechar, string)
fullstring = string
IF position > 0
   string = LEFT(fullstring, position - 1) + LTRIM(STR(pagenum))
   IF position < LEN(fullstring) - LEN(pagechar) + 1
      string = string + SUBSTR(fullstring, position + LEN(pagechar))
   ENDIF
ENDIF
RETURN
* EOP Getpage
```

Special Thanks

Owing to the unusual nature of the snaked column report, a number of different approaches to the problem surfaced. Although the article here presents just one of the many techniques explored, a number of people contributed to the discussion and so deserve credit for keeping the idea alive. A special thanks to

Roger Wegehoft
Vince Mendillo
Robert Boies
Tim Lebel
Kenneth Getz
Olivier Biggerstaff
Christopher White

dBASE TIPS

By J. Michael Hedblom

dBASE III PLUS

@...PICTURE

When you use @...PICTURE with negative numbers, you need a template position for the minus sign. If you don't have one and the number fills the number of template digits you define, an overflow will display. Including the "@(" PICTURE function to surround negative values with parentheses also requires a position for the minus sign, although the symbol is not used. The length of the template must be at least the length of the number plus one for the sign. For example,

```
mvar = -100
@ 5,20 SAY mvar PICTURE "999"
@ 6,20 SAY mvar PICTURE "@( 999"
```

each display an overflow. However,

```
@ 5,20 SAY mvar PICTURE "9999"
@ 6,20 SAY mvar PICTURE "@( 9999"
```

correctly display, "-100" and "(100)," respectively.

ASSIST

There is no way to close files from ASSIST. To close files opened in ASSIST, select ASSIST **Set Up:Quit dBASE III**, or execute a CLEAR ALL or appropriate CLOSE command from the dot prompt after leaving ASSIST.

INSTALL DBA

The following steps install dBASE Administrator version 1.1 when the dBASE III PLUS System Disk #1 is unusable for any reason.

1. Set the default directory on the destination drive to the Administrator directory. For example, assume the destination drive is C:

   ```
   CD C:\DBA
   ```

2. Insert Administrator Disk #1 in drive A: and enter

   ```
   A:
   IDLAN
   INSTALLH P=A T=C
   COPY DBA.LOD C:\DBNETCTL.300/v
   COPY DBASE.MSG C:/v
   COPY PROTECT.EXE C:/v
   COPY ASSIST.HLP C:/v
   ```

3. Insert Administrator Disk #2 in drive A: and enter

```
COPY ADDUSER.COM C:/v
COPY DBA.OVL C:/v
```

4. Insert System Disk #2 in drive A: and enter

```
COPY HELP.DBS C:/v
```

During a normal installation, HELP.DBS is copied from System Disk #1. If System Disk #1 is damaged, INSTALL.BAT aborts before the Administrator disks can be installed. The HELP.DBS file on System Disk #2 does not contain commands and functions specific to the dBASE Administrator in it.

You should only use these steps if the System Disk #1 is damaged and not in place of normal installation procedures.

SET CATALOG ON

With an active Catalog and SET CATALOG ON, creating a new index or issuing SET INDEX TO an existing index file does not prompt for a file title description as documented in *Using dBASE III PLUS*. Instead, dBASE III PLUS places the index key expression in the Catalog field Title.

SET COLOR TO

Using an attribute (high intensity or blinking) of colors with SET COLOR TO applies to the foreground setting only and has no effect on background colors. In fact, if you assign an attribute to a background color, dBASE III PLUS passes the attribute to the foreground color instead. Because of this, the colors gray (N +) and yellow (GR +) appear as black and brown respectively when used for background settings.

Keep in mind that for color settings there are three basic settings (display, enhanced, and border) and that within each one there are both foreground and background colors. The general syntax is

```
SET COLOR TO [[<standard>][,<enhanced>][,<border>]]
```

where

```
standard  ::= [<foreground>][/<background>]
enhanced  ::= [<foreground>][/<background>]
border    ::= [<foreground>][/<background>]
```

and

```
foreground ::= [<letter>[{attribute}]]
background ::= <letter>
```

It appears that the attribute, although it affects the color, is a function of the setting. Each setting, therefore, can have only one attribute which is applied to the foreground color. This behavior is in contrast to dBASE III version 1.0 and 1.1, where attempting to assign an attribute to a background color did not pass the attribute to the foreground color.

SET FILTER TO

It is not documented in any of the dBASE III PLUS reference manuals that certain commands ignore the SET FILTER TO condition. Specifically, the commands DISPLAY (current record), DISPLAY RECORD <n>, GO/GOTO, or any command with the scope RECORD <n> or NEXT <n>. For example, if the SET FILTER TO condition excludes record five and GOTO 5 is executed, a subsequent DISPLAY reveals the contents of the record.

STORE

Attempting to STORE a string containing "&&" to a memory variable produces the error message "Unterminated string" when executed. This is because the double ampersands are interpreted as an in-line comment, effectively removing the closing quotation mark from the command line. To work around this, split the character expression between the ampersands into two expressions and concatenate them when assigning them to the memory variable. For example, the following is a successful instance of embedding "&&" in a string.

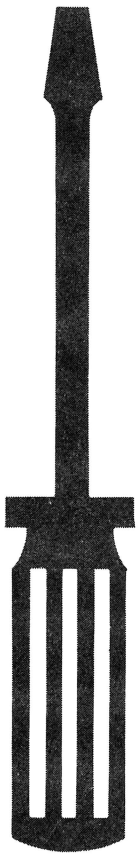
```
STORE "Before the &" + "& and after." TO mvar
```

UNINSTAL.BAT

UNINSTAL.BAT deletes CONFIG.SYS, if it exists, in the target drive default directory. This is potentially dangerous if the default directory of the target disk is the root directory. Some systems, such as older Bernoulli drives, require CONFIG.SYS to boot. Use caution when uninstalling dBASE III PLUS from a root directory.

Using the *Framework II Developer's Toolkit* with Microsoft C

BY KENNETH N. GETZ



The newly released *Framework II Developer's Toolkit* gives you the opportunity to add many functions to Framework II that were not previously available. In addition to printer and plotter driver construction kits, a library of user-defined functions, and direct access to dBASE III PLUS database files, the Toolkit provides information on the Assembly Language Interface to Framework II that was previously undocumented. This interface allows you to control the desktop more closely and to develop applications that would otherwise have been relegated to other languages or environments.

All of the documentation and example programs are, however, written for use with the Lattice C compiler. To use the Microsoft C Compiler (version 3.0 or higher), you will need to make some adjustments to those programs or you'll be faced with some unpleasant "Unresolved external" error messages when LINKing your applications.

There are always differences in compilers, and one place for such differences is in the machine-language interface. This interface is where most code changes need to be made. Very few sections of the C routines will need to be changed. We'll document those changes here and demonstrate the (slightly) different compiling instructions so that users of Microsoft C can write Framework-callable routines as well. Because these guidelines are so general, you should consult your Microsoft C manual for further information on compiling and linking.

We've also included some example C programs and their calling frames for adding Month, Day, and Year functions to Framework II. These examples are designed to help you get started. The information for each of them was drawn from the example programs included with the Toolkit.

Changes to Mymain.ASM

Precede All Public Variable and Segment Names with an Underscore

Microsoft C requires that all public variables and segment names begin with an underscore. Therefore, you must modify Mymain.ASM (and any other .ASM files you use with C programs for Framework II), adding an underscore to every procedure that the C program will access. You must repeat this also for segment names. For example, the Data segment must be referred to as "_DATA", and the Stack segment as "_STACK." Without this, you will get a series of "Unresolved external" error messages when you link.

Change the Code Segment and Class Names

The Microsoft C compiler expects the name of the code segment to be "_TEXT". You must, therefore, change any references to the group name "PROG" in Mymain.ASM to "_TEXT." Microsoft also assumes the class name to be "CODE." You should change the segment directive line to read:

```
_TEXT SEGMENT BYTE PUBLIC 'CODE'
```

Add Declarations for Necessary Segments

Your main assembly language program (Mymain.ASM for the example program, Readtext.C) requires references to two extra segments for it to work correctly with the Microsoft C Compiler. To correctly initialize these segments in order, add the following block of declarations to the top of your main .ASM program:

```
_DATA    SEGMENT WORD    PUBLIC    'DATA'
_DATA    ENDS
CONST    SEGMENT WORD    PUBLIC    'CONST'
CONST    ENDS
_BSS     SEGMENT WORD    PUBLIC    'BSS'
_BSS     ENDS
```

Also change the declaration for DGROUP to read:

```
DGROUP    GROUP    CONST,  _BSS,  _DATA,  _STACK
```

Add Reference to_acrtused

You must link your program without including the default startup code, since this module is used to set up the stack, registers, and command line parameters that **@exec** handles for you. The Microsoft compiler includes a reference to a variable named "_acrtused" that forces the inclusion of the startup code at link time. It appears that this PUBLIC reference is used *only* to force the compiler to link in the module CRT0. CRT0 has only a single use of "_acrtused," which is assigned a value of 9876H and is commented to be "a funny value not easily matched in SYMDEB." Declaring a similar PUBLIC constant in the data area of your main assembly-language calling program will take care of this otherwise unresolved external reference when linking.

Preserve SI and DI Registers

Microsoft C presumes that the SI and DI registers have been preserved after any calls to external routines. To ensure this, PUSH and POP the SI and DI registers in any assembly language routine that uses them. As a general practice, save and restore these registers in all assembly language routines you write.

Summary of Changes to Mymain.ASM

The following program is a "skeletonized" version of the example Mymain.ASM found in the \LLI\RDTXT directory of the ***Framework II Developer's Toolkit*** Disk #2. Compare this version to that found on the diskette in order to see the changes you need to make.

```
; Mymain.ASM  (Rewritten for Microsoft C version 4.0)
;
; Please refer to the example Mymain.ASM given in the directory
; \LLI\RDTXT on diskette #2 for comparison to this skeleton of the program.
;
; Notice the typographical error on the next line.  The original version
; has an underscore (init_proc) while the C routine does not.
;
        extrn   _initproc:near
        extrn   _main:near
        extrn   _wrap_up:near
_DATA   segment word    PUBLIC  'DATA'
_DATA   ends
CONST   segment word    PUBLIC  'CONST'
CONST   ends
_BSS    segment word    PUBLIC  'BSS'
_BSS    ends
DGROUP  GROUP   CONST, _BSS, _DATA,_STACK
_DATA   segment word    PUBLIC  'DATA'
PUBLIC FW_EXEC          ; No underscore, since never referenced from a C
fw_exec dd      0       ; module.
; The entry point array doesn't change and goes here.
; Add the following lines for Microsoft C.
PUBLIC  __acrtused
__acrtused = 9876h      ; The actual value probably doesn't matter.
_DATA   ends
_STACK  segment word STACK 'DATA'
        dw 1024 dup (?) ; declare stack for local usage
_STACK  ends
PGROUP  GROUP _TEXT
_TEXT   SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS:_TEXT,DS:DGROUP,ES:DGROUP,SS:DGROUP
init_procf      proc    far
        call    _initproc
        ret
init_procf      endp

mainf   proc    far
        call    _main
        ret
mainf   endp
```

```
wrap_upf        proc    far
        call    _wrap_up
        ret
wrap_upf        endp
; For each of the C-callable routines following, precede each
; reference to a procedure with an underscore, both in the PUBLIC
; declaration and in the PROC beginning and ending.
; All the procedures go here with underscores before PUBLIC proc names, as
; in the following example.
public _fw_read
_fw_read        proc    near
; The internals of the procedures do not change.
_fw_read        endp
;
; The rest of the C-callable routines go here, changes as above.
; The following doesn't change.
;
dummy_main      proc    near
        ret
dummy_main      endp
;
; And end the segment (with the underscore!)
_TEXT   ends
        end     dummy_main
```

Changes to the Toolkit C programs

Most of the C programs in the Toolkit should require no changes in order to run. You will, however, need to make a small change to Sutils.C (used in the Readtext.C example) to be able to use it with Microsoft C. Specifically, within the body of the function, change the line that reads:

```
temp = (unsigned *) *str;
```

to

```
temp = (unsigned) *str;
```

This change will allow the module to compile correctly.

Compiling and Linking

Use LINK Version 3.51

It is in the linking process where you are likely to encounter the most trouble. Be sure to use Microsoft LINK Version 3.51 (or later if released). This is the linker that comes with Microsoft C Version 4.0 and is downwardly compatible with other versions of LINK.

Compile and Link Command Line Options

When you assemble, compile, and link, MASM.EXE, LINK.EXE, and all files necessary to compile the C program must be in the current directory or in your DOS path. In addition, all environment variables must have been correctly set up. See the *Microsoft C Compiler User's Guide*, Chapter 2, for details on setting up. The following series of command lines shows the order of DOS command lines with options to create an .EXE file you can bring to the Framework II desktop:

```
MASM Mymain;
MSC <your C program> /Gs;
LINK Mymain + <your C program>, <your C program> /NOD;
```

The "/Gs" option on the Microsoft C compile command line disables stack checking and the "/NOD" on the LINK command line suppresses the inclusion of the default libraries. You must compile without stack checking, since the stack checking code assumes that its C code has set up the stack. In this case, Framework II is handling the stack.

Although the documentation for the Toolkit recommends avoiding compiler libraries, many of the standard library functions will work. If you decide to use library functions, leave off the "/NOD" from the LINK command. Be aware that the use of these libraries is not recommended by the Toolkit authors and not all library routines will function correctly in the Framework II environment.

An Example Program

To understand how the process of developing a user-defined function with the Toolkit works, review the following example application. Here you will create a C program (Day.C) that can convert a Framework II date into a number representing either the day, month, or year. You are then shown how to take this code through the various phases of development to create the FRED user-defined functions, @Day, @Month, and @Year.

The process is not very difficult and consists of the following general steps:

1. Create the C source program.
2. Modify Mymain.ASM to create a header program for the C program.
3. Compile, assemble, and link the object code files together to form an executable (.EXE) file.
4. Load the .EXE file to the Framework II desktop and merge it into a FRED program or user-defined function.
5. Save it as a Framework II frame.

Create the C Program

Create the following program, Day.C as an ASCII file. You can do this either in an editor that supports ASCII files directly or from within Framework II by exporting to an ASCII file using **Disk:Export:ASCII Text**. If you create and export from Framework II, be sure to rename the source file from Day.TXT to Day.C.

```
/* Program ...: Day.C */
/* Author ....: Kenneth N. Getz */
/* Date ......: June 1, 1987 */
/* Note(s) ...: To be linked with Dates.OBJ and called from Framework II. */
struct data
   {
   int type;
   unsigned char date[8];
   };

int day()
{
   struct data value_rec;

   GET_VAL(1,&value_rec);      /* Get the value from the first frame. */
   if (value_rec.type != 2)    /* If the parameter isn't a date type. */
      return(-1);
   return(value_rec.date[3]);
}

int month()
{
   struct data value_rec;
   GET_VAL(1,&value_rec);      /* Get the value from the first frame. */
   if (value_rec.type != 2)    /* If the parameter isn't a date type. */
      return(-1);
   return(value_rec.date[2]);
}

int year()
{
   struct data value_rec;
   GET_VAL(1,&value_rec);      /* Get the value from the first frame. */
   if (value_rec.type != 2)    /* If the parameter isn't a date type. */
      return(-1);
   return(value_rec.date[0]+(value_rec.date[1] * 256));
}
void main()
/* This is a "dummy" main, used only to satisfy the compiler. */
{
   return;
}
```

Create the .ASM Header Program

To create the necessary assembly-language interface header for Day.C, make a DOS copy of Mymain.ASM file found in the \LLI\RDTXT directory of the Toolkit Disk #2 to Dates.ASM. Then modify Dates.ASM as specified in the listing below. Note that only the top section of Dates.ASM needs to be changed.

```
; Dates.ASM
; This program is the "front-end" for the C program Day.C
```

```
;
   extrn _day:near
   extrn _month:near
   extrn _year:near
_DATA SEGMENT  WORD  PUBLIC   'DATA'
_DATA ENDS
CONST SEGMENT  WORD  PUBLIC   'CONST'
CONST ENDS
_BSS  SEGMENT  WORD  PUBLIC   'BSS'
_BSS  ENDS
DGROUP   GROUP CONST, _BSS, _DATA,_STACK
_DATA segment word PUBLIC 'DATA'
PUBLIC FW_EXEC
fw_exec  dd 0            ; Will be plugged by Framework II.
;
; The entry point array
proc1 dw offset day_f    ; Offset addr of proc # 1.
proc2 dw offset month_f  ; Offset addr of proc # 2.
proc3 dw offset year_f   ; Offset addr of proc # 3.
dummy_buffer   dw 0      ; Used by num_bytes.
public   __acrtused      ; Trick to force in startup.
__acrtused = 9876h       ; Funny value not easily matched in SYMDEB.
_DATA ends
_STACK   segment word stack 'DATA'
   dw 1024 dup (?)       ; Declare stack for local usage.
_STACK   ends
;
; Segmentation set up to reflect the SMALL model as implemented
; by various "C" compilers.
PGROUP   GROUP _TEXT
_TEXT SEGMENT  BYTE  PUBLIC   'CODE'
ASSUME   CS:_TEXT,DS:DGROUP,ES:DGROUP,SS:DGROUP
;
; The following 3 procedures are the FAR entry points. Their
; only purpose is to call the real (near) routines which are
; written in "C."
day_f proc  far
   call  _day
   ret
day_f endp
month_f  proc  far
   call  _month
   ret
month_f  endp
year_f   proc  far
   call  _year
   ret
year_f   endp
;
; The code following this point does not need any changes other
; than the ones previously outlined in this article.
```

Compile, Assemble, and Link

Once you have created the source files, execute the following commands to create the final executable file. Before you begin, be sure that your DOS path and environmental variables have been set as indicated in Chapter 2 of the *Microsoft C Compiler User's Guide*. The command lines below show the steps to follow. Do not type the comments surrounded by "/*" and "*/".

```
MASM Dates;     /* Assemble Dates.ASM into DATES.OBJ */
MSC Day /Gs;    /* Compile Day.C, disabling stack check */
```

Now link DATES.OBJ and DAY.OBJ to form DATES.EXE. Remember to not include any default libraries.

```
LINK Dates + Day, Dates /NOD;
```

Setting Up the FRED User-Defined Functions

Now that you have created DATES.EXE, you are now ready to create the FRED user-defined functions Day, Month, and Year. To begin, create an empty word frame and label it Dates. Then **Dnlevel** and under Dates create four empty subframes labeling them as follows:

Dates
- Date
- Day
- Month
- Year

In the formula area of Day, Month, and Year, **EDIT FORMULA (F2)** and enter the following formulas:

```
; Day.FW2
;
;
; Called as : @Day(<date>)
; Returns   : Day as an integer,
;             -1 if not date type.
;
date := @item1,
@execparms("date"),
@return(@exec(Getdate, 1))
```

```
; Month.FW2
;
;
; Called as : @Month(<date>)
; Returns   : Month as an integer;
;             -1 if not date type.
;
date := @item1,
@execparms("date"),
@return(@exec(Getdate, 2))
```

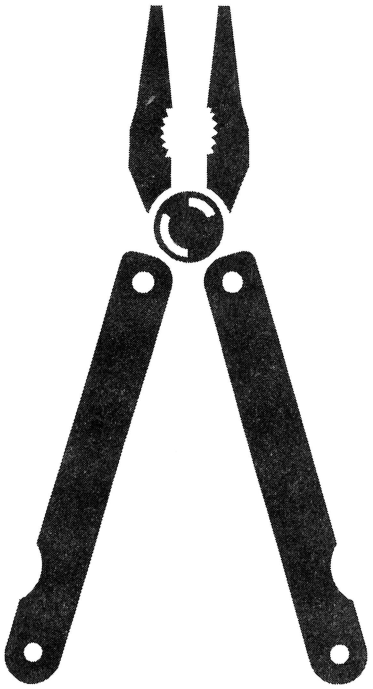
```
; Year.FW2
;
;
; Called as : @Year(<date>)
; Returns   : year as an integer,
;             -1 if not date type.
;
date := @item1,
@execparms("date"),
@return(@exec(Getdate, 3))
```

Now **Disk:Get File by Name** DATES.EXE, rename it to Getdate, and **MOVE (F7)** it under Dates. Your frame structure should now look like this:

Dates
- Date
- Day
- Month
- Year
- Getdate (originally DATES.EXE)

To make references to Day, Month, or Year global, **MOVE (F7)** Dates under the User-defined functions section of your Library. Day, Month and Year can now be called, just like any internal Framework II functions.

Special thanks to Bert Kleinman of Pacific Palisades, California, for his help uncovering the methods to interface Microsoft C Version 4.0 with the Framework II Developer's Toolkit.



References

1. Microsoft Corporation. *Microsoft Macro Assembler User's Guide*. Microsoft Corporation, 1985. pps. 53,61.
2. Microsoft Corporation. *Microsoft C Compiler User's Guide*. Microsoft Corporation, 1986. pps. 17–48, 213–221.
3. Prata, Stephen. *Advanced C Primer + +*. Indianapolis: Howard W. Sams & Co., 1986. pps. 361–413.

Translating FRED Programs from One Language to Another

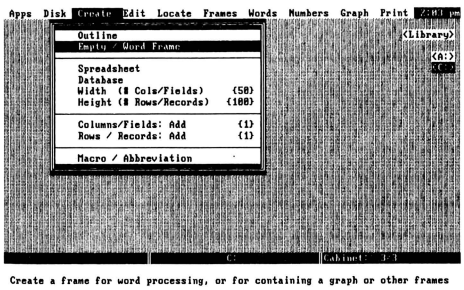
BY PAUL BUKAUSKAS

The Framework II programming language, FRED, is basically the same among the English and foreign language versions. The only differences are the spellings of the menu options and the functions that access them, such as **@performkeys** and **@sense**. When you translate a FRED program from one language to another, you need only change the keystrokes used to access these menu options. The translation examples below are from English to Spanish, although the concepts are true for any language.

Before translating Framework II programs from one language to another, first be sure the program works correctly. Next, work with a copy of the original program, so you have an unmodified copy as a backup.

@performkeys

The **@performkeys** function is used to execute keystrokes automatically— to select menu options within a program, for example. Translating **@performkeys** requires that you change the letters used to invoke the menu options. For example, in the English versions of Framework II you use the statement, **@performkeys**("{Ctrl-C}E") to create an empty word frame. In the Spanish versions, you use **@performkeys**("{Ctrl-C}V"). Figure 1 shows you the create menu in the English and Spanish versions of Framework II.



(a) Create menu in the U.S. version. (b) Create menu in the Spanish version.

@sense

The **@sense** function is used to evaluate the status of a toggled option in one of the pull-down menus. Like **@performkeys**, it uses the first letter of the menu name and option as an argument. For example, in the English version, **@sense**(#ET) evaluates whether **Edit:Typeover** is on or off. In the Spanish version the syntax for the same action is **@sense**(#CD).

To help you translate **@sense** from one language to another, refer to the FRED language section of the *Framework II Advanced Topics* manual for a list of acronyms used by **@sense**.

The table below lists the acronyms used by **@sense** in the English and Spanish versions of Framework II.

Table of Constants (English/Spanish)

Apps Menu	Menu Util
#ASH	#UDV
#ASD	#UDD
#ATTUW	#UCTTE
#ATTUH	#UCTTG
#ATTUR	#UCTTS
#ATTDC	#UCTRE
#ATTDF	#UCTRG
#ATFE	#UCFA
#ATSE	#UCPE
#ATST	#UCPT
#ATSSFB	#UCPOFV
#ATSSFD	#UCPOFN
#ATSSFP	#UCPOFP
#ATSSFS	#UCPOFB
#ATSSMD	#UCPOME
#ATSSMS	#UCPOMN
#ATSSCR	#UCPOCT
#ATSSCS	#UCPOCE
#ATSSCD	#UCPOCD
#ATST	#UCPTT

Edit Menu	Menu Editar
#EL	#EB
#ET	#ED
#ED	#EM

Locate Menu	Menu Local
#LL	#LE
#LC	#LC
#LF	#LF
#LI	#LI

Graph Menu	Menu Gráfico
#GOM	#GOI

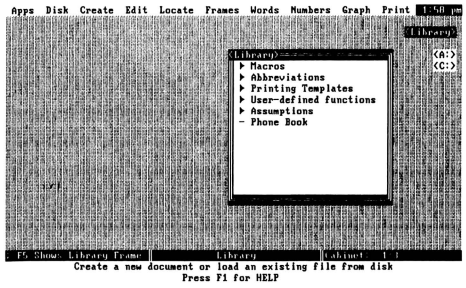
Frames Menu	Menu Ventana
#FV	#VM
#FN	#VN
#FL	#VR
#FD	#VV
#FH	#VO

Print Menu	Menu Imprimir
#PP	#IP
#PW	#IR
#PFPC	#IVFT
#PFPQ	#IVFE
#PCK	#III
#PCB	#IIE
#PDD	#IDF
#POS	#IOO
#POF	#IOF
#POP	#IOI

The Library Cabinet

Since the term Library is language specific, the name used to refer to the Library Cabinet varies from one language version to another. This means that if you make reference directly to the Library Cabinet, you need to change the spelling of the Library to the language version you are using. For example,

@setselection("Library.Phone Book")



would be

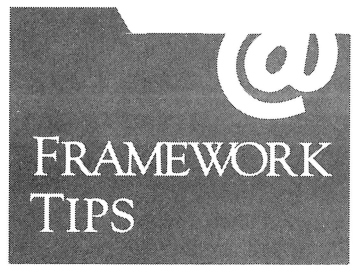
@setselection("Libreria.Agenda telefónica")

in the Spanish version.

Print:Format Options

If your program uses the <even/odd> or <page> options in the **Print:Format Options:Headers** or **Print:Format Options:Footers** options, change the spellings so that they are correct for the particular language.

This is an example of the kinds of changes you have to make when translating a FRED program from one language to another. Although it does not affect the execution of a program, you may want to translate the messages and prompts that display on the screen so that they display in the appropriate language. You may also want to change the spelling of frame names—just be sure to also make these changes to the functions that access those frames.



BY CHRIS LO VASCO

@setselection

If you have **@setselection** in a spreadsheet cell, it always returns the value "<spreadsheet path>.A1" during a full recalculation from within the spreadsheet. If you initiate a full recalculation from a location outside the spreadsheet, the value returned is the reference path to the current cursor location.

There are two instances when the name of the current cell is correctly returned: when you perform a local recalculation by setting **Numbers:Options for Recalc:Manual** on, press **EDIT FORMULA (F2)**, and then **Return** on the cell; when you use **@setselection** to reference another cell in the spreadsheet.

Cursor Pointing to Copy Displayed Values Only

The # to end cursor pointing is commonly thought of as picking up the value in the indicated cell, rather than the formula. It actually picks up the displayed value and this distinction is important. When you cursor point, the displayed value of the pointed-to cell is entered into the current formula exactly as it appears. The result of entering the displayed format of a value into a formula may surprise you before you think it through completely.

For example, numbers in the Business format may include commas, but FRED interprets commas as delimiting a list of numbers. Thus if you cursor-point to a number displayed with commas, then the number will be entered in the formula with its commas, and the value displayed for the formula will be only the digits following the last comma. For example, if the cell you cursor-point to displays 34,567, then your current cell will contain the formula 34,567 and a displayed value of 567. (The displayed value may look like 567.00 or 56700.00%, and so on, depending on the current cell's display format.)

Here's a second example. Suppose you cursor point to a cell in Scientific format whose displayed value is 3.46E + 4, and your current cell is in Business format. The result is the formula 3.46E + 4 and the displayed value 34600.

Thanks to Marilyn McMahon of Forefront for this explanation.

Maximum Number of Drive Cabinets

Framework II allows 26 floppy drives and 26 hard drives to be configured using Setup. However, due to a limitation in Framework II, only 9 drive cabinets and the Library cabinet can be displayed. Framework II displays only drive cabinets A through J.

This happens in both versions 1.0 and 1.1 of Framework II.

Creating Sized Frames

The macro, @Sizeto, allows you to create a frame at a certain position and of a certain size from within a FRED program. You can place it in the Library Cabinet and call it from any program that needs to create a frame "on the fly" during program execution.

This macro first creates a frame, then expands it to the largest possible size by performing **ZOOM (F4)** twice. Then, the **@panel** functions are used to record the maximum number of rows and columns into the frames Max Rows and Max Cols. These are stored as frames, rather than as local variables, so that other programs can access them if they need to determine the screen dimensions. The macro uses **DRAG (F3)** to move the upper left-hand corner of the frame and then **SIZE (F4)** to move the lower right-hand corner to the coordinates passed as parameters. @Sizeto inserts the name you sent as a parameter in the frame label and returns control to the calling program. Since this macro was written to be used as a utility called from other programs, it does not control the status of **@echo**. You must set **@echo** #OFF yourself before the call to @Sizeto and set it #ON after the call if you do not want the frame movement to appear on screen.

To setup @Sizeto, create a word frame in the User-Defined Function area of your Library frame named Sizeto, and two frames in the Assumptions area labelled Max Rows and Max Cols. Next, enter the program code into the formula area of Sizeto. Leave Max Rows and Max Cols empty, as they will be filled in the first time that @Sizeto is called.

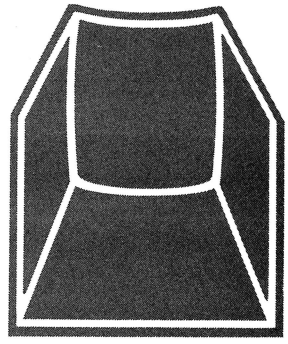
To use @Sizeto, call it with this syntax: @Sizeto(" < framename > ",
< upper left row > , < upper left col > ,
< lower right row > , < lower right col >

If the frame was formed correctly, the function returns 0. If there was an error (caused by incorrect dimensions sent as parameters), the function returns a -1.

For example, to create a frame named "Test" whose upper left-hand corner was at location (3,5) on the desktop and whose lower right-hand corner was at location (10,15) on the desktop, use this syntax: @Sizeto("Test",3,5,10,15)

```
; Sizeto
;
;    by Kenneth N. Getz
;       June 1, 1987
;
; Used to create a frame at a specified location on the screen.
;
; @Sizeto(<frame name>, <upper left row>, <upper left col>,
;             <lower right row>, <lower right col>)
;
; Returns 0 if frame formed OK, -1 if there as an error.
```

```
;
; Make sure we're at the desktop level.
@performkeys("{Ctrl-Uplevel}"),
;
; Create the new frame and zoom to full screen, storing away the maximum
; screen coordinates.  Verify that the parameters passed conform to the
; possible position values.
;
@performkeys("{Ctrl-C}E{F4}{F4}"),
Max rows := @panel2,
Max cols := @panel1,
@if(@or(@item2 < 1, @item3 < 1, @item4 > Max Rows +2, @item5 > Max Cols +2),
     @list(
          @performkeys("{RETURN}{DEL}"),
          @return(-1)
     )
),
; Perform DRAG (F3) to move upper left corner.
@performkeys("{F3}"),
@performkeys(@rept("{Ctrl-Dnarrow}", @int((@item2 - 1)/4)) &
               @rept("{Dnarrow}", @mod(@item2 - 1, 4)) &
               @rept("{Ctrl-Rightarrow}", @int((@item3 - 1)/8)) &
               @rept("{Rightarrow}", @mod(@item3 - 1, 8)) &
               "{Return}"
),
;
; Perform SIZE (F4) to move lower right corner.
@performkeys("{F4}" &
               @rept("{Ctrl-Uparrow}", @int((Max rows + 2 - @item4)/4)) &
               @rept("{Uparrow}", @mod(Max rows + 2 - @item4, 4)) &
               @rept("{Ctrl-Leftarrow}", @int((Max cols + 2 - @item5)/8)) &
               @rept("{Leftarrow}", @mod((Max cols + 2 - @item5), 8)) &
               "{Return}"
),
;
; Put the name in.
@performkeys(@item1 & "{Return}"),
@return(0)
```



Automating Applications Using Macros

BY RICHARD J. AUGELLI

Key macros can save keystrokes when you perform repetitive tasks such as typing frequently used words or executing a series of related menu options. In total, they take less time to execute than if you enter keystrokes manually, have no typing errors, and above all free you from the mental burden of remembering a complex set of related tasks. Inevitably, you are more productive and your work is more pleasant. In this article, we will discuss key macros and how they can be used in RapidFile. Then we will apply this knowledge to create a boilerplate text library useful for automated letter and memo writing.

What are Key Macros?

Key macros in RapidFile are like key macros in many programs. They are a series of keystrokes assigned to a single key so that when you press the macro key, a stored set of keystrokes are executed in place of the macro key. In RapidFile, there are two ways to create them. You can assign the keystrokes to a macro key by recording them as you execute the task you want to assign to the macro key or you can enter the keystrokes into the macro editor without executing any actions. Both methods have merit, but it is generally best to begin by recording and then fine-tuning your key macros in the macro editor later.

Creating and Editing Key Macros

To create a key macro in RapidFile, press **Ctrl-K** to begin recording keystrokes. When you hear a midrange tone, name the macro by entering a key combination, such as **Alt-F2**. Then when you hear a high-pitched tone start entering the keystrokes you want to record. As you enter the key sequence you'll hear a short beep. After you perform the last keystroke you want to include in the key macro, press **Ctrl-K** or **Ctrl-Break** to stop recording. Be aware that when you create a key macro, RapidFile does not check to see if the macro key has already been assigned, making it very easy to overwrite an existing macro key definition.

As mentioned above, you can also view and edit your key macros in the macro editor. When you press **Ctrl-E**, an Edit Macro screen appears showing the macros in alphabetical order starting with **Alt-A**. You can then scroll down by using **Dnarrow** or **PgDn** until you arrive at the macro key definition you want to edit. Press **EDIT (F2)** and the macro is displayed using different symbols for each key. These symbols are translated at the bottom of the screen. By moving the cursor, you can insert or delete keystrokes. Once you are done editing your macro keys, **IMPLEMENT (F9)** to save or **Esc** to exit the Edit Macro screen.

An additional capability of key macros in RapidFile is that you can nest them. This allows you to include previously defined key macros within the key sequence of another key macro.

Types of Key Macros

In RapidFile there are three types of key macros: **Memo Writer, Data File,** and **Global**, each working in a specific environment and stored in a different file.

Memo Writer key macros are defined as the combination of the **Alt** key with any letter. These key macros are saved in a special DOS file called MEMO.MAC and are global to all memo files. MEMO.MAC files can be copied and renamed like any regular DOS file making it possible to create a library with different sets of Memo Writer key macros.

For example, you could create a key macro **Alt-T** that sets and clears the tab stops. This key macro performs a **Ctrl-LT**, acting as a toggle to set or clear a tab stop. Other very useful key macros are those that create forms "on the fly." With a two-key combination you can create a memo, a form letter, or any kind of standard document, freeing you from the burden of typing it every time.

Data File key macros can only be called within the database file where they are defined. They also use a combination of **Alt** and any letter. Typically these are directly related to the set of tasks you have defined for the current database file and tend to be application-oriented. Automated data entry is a good example of the type of task you would assign to this kind of macro key.

As an example, suppose you have a database file in which you enter your checkbook expenses. Some expenses are for the same amount, such as insurance premiums, mortgage payments, or rent. By creating a key macro, you can input these items just by pressing a macro key combination. If the current date has to be entered, it can also be defined in the key macro as **Ctrl-/** to insert today's date into the appropriate field.

Another useful application of data file key macros is partial editing of fields in the database. Suppose you did not remember to enter the area code in your phone number field. A key macro can be used to **EDIT (F2)**, insert the area code, **PgDn** to close the edit panel, and position the pointer on the next record.

Global key macros are the last type of key macros. They are referred to as global because they can be invoked from any part of RapidFile (any database in the file manager or any document in the Memo Writer). They are named using a combination of either the **Ctrl**, **Shift**, or **Alt** keys and any function key. Like the other types of key macros, they are saved in a DOS file (called RF.MAC).

Typically, global key macros are used for menu navigation and applications that move between the file manager and the memo writer. As an example of enhanced menu navigation, you might want to define global key macros to **File:Disk/Directory:Access DOS** or **Layout:Snapshot:Use Snapshot.**

A Boilerplate Library

The following application is a boilerplate text library that demonstrates how you can use key macros to build a letter from a database file containing letterheads, paragraphs, and form letters. Basically, it works by appending the contents of a field or its linked memo file to a "clipboard" memo file. Another key macro copies the contents of the clipboard to the memo you are creating and then clears the clipboard.

To set this up, create a database file with three text fields: "Letterheads," "Paragraphs," and "Letters." Each cell of the database file will contain either a letterhead address, a standard paragraph, or a form letter. When you enter information into a cell, first enter a name that identifies that particular paragraph, such as "Salutation" or "Recommendation" and then **ZOOM (F4)** into the memo writer. The first line in the memo writer will show the title of the paragraph. Skip one line, enter the desired paragraph and **ZOOM (F4)**. To test that the key macros correctly enter several records into each one of the fields, check to see that the database file looks like Figure 1.

```
File      Layout      Records      Write      Print                    12 34 pm

     LETTERHEADS     PARAGRAPHS      LETTERS

  1  COMPANY1        SALUTATION      SALES
  2  COMPANY2        IMPORTANCE      WELCOME
  3  COMPANY3        COLLECTION      REFERENCE
  4  COMPANY4        CLOSING         RECOMMENDATION

TABLE: Unsaved layout  ‖ G:BOILER.RPD ‖ Selected:     4 of 4       ‖
          F1 Help    F3 Flip    F5 Tag          F7 Move    F9 Implement
          F2 Edit    F4 Zoom    F6 Highlight    F8 Copy    F0 Menu
```

Figure 1

Next, create global key macros from the file manager. Name the first key macro **Alt-F2** and enter the following key sequence:

{F4}{Dnarrow}{F6}{Ctrl-End}{Return}{Ctrl-F}SHCLIPBRD{Return}A{F4}

This key macro now appends the different paragraphs you select to a memo file called CLIPBRD.MMO. Keep in mind that **Alt-F2** assumes the memo text is entered on the second line. It is, therefore, important that you enter the paragraphs leaving a blank line after the title. The macro also expects a blank line at the end of each paragraph so that the final letter contains blank lines between paragraphs.

The second global key macro is **Alt-F3** and consists of the following keystrokes:

{Ctrl-F}ICLIPBRD.MMO{Return}Y{Ctrl-F}DECLIPBRD.MMO{Return}Y{Esc}{Esc}{Ctrl-End}

Alt-F3 copies whatever was appended to the clipboard into your current memo. The combination of these two key macros allows you to append a number of paragraphs or an entire letter to a document with previously entered information.

You are now ready to test the application. Highlight the name of the letterhead, paragraph, or form letter you want to include in your letter and press **Alt-F2**. The key macro takes care of the rest, copying the selected text to the clipboard. When the key macro completes, highlight the next paragraph you want to include in the letter and press **Alt-F2** again. Continue to select and cut paragraphs to the clipboard. When you finish, create a memo file (**Ctrl-W**) and press **Alt-F3** to paste the contents of the clipboard into the current document and clear the clipboard. Additional text can be entered either before pasting the contents of the clipboard into the memo file or after some text has been pasted. In addition, a form letter created with these two key macros can be linked to a field in the database. You can do this by **ZOOM**ing **(F4)** into the memo writer and linking the memo file to the field if necessary.

To enhance and make the application more user friendly, add an opening menu that comes up when RapidFile is loaded. To do this, create a database file with just one text field called Menu. Set **Fields:Width of field** to 42 and **Fields:Height of field** to 3. Then enter the text options as you see in Figure 2 below.

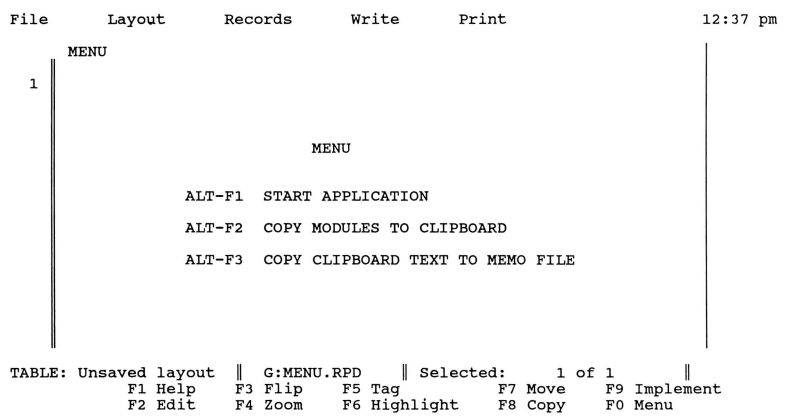


Figure 2

Name this database file MENU.RPD and **File:Save** it. Then to make the menu come up whenever you load RapidFile, use "File to load on startup" option of the file manager settings in RF CUSTOM to retrieve MENU.RPD. Last, create the **Alt-F1** key macro to start the application by entering the following key sequence:

{Ctrl-F}RMENU

With this convenient enhancement, RapidFile displays the menu when it's loaded and will load the boilerplate text library when you press **Alt-F1**. This is a good addition if your application is going to be used by inexperienced users.

Press **Ctrl-E** and **IMPLEMENT (F9)** to save the global key macros to disk before exiting RapidFile.

This article has given you a quick overview of key macros in RapidFile with a sample application. For further information, refer to chapter 19 of your *Using RapidFile* reference manual. For an expanded set of applications examples, see Ashton-Tate's *RapidFile Business Applications* by Leo Brodie. This book is an excellent reference for ideas and techniques to build applications in RapidFile.

References

1. Brodie, Leo. *RapidFile Business Applications*. Torrance: Ashton-Tate, 1987. pps. 96, 215–217.

RapidFile Tips

By Scott Haynes

Dividing by Zero

If you attempt to divide by zero in a calculated field, RapidFile returns asterisks as the field value. If you subsequently address a field with this value with another calculated field RapidFile returns asterisks into this second calculated field.

Report fields treat these calculated fields as zero.

File to Load on Startup

Drive letters and directory names are not allowed in the **File to load on startup** option in RF Custom. If you enter them, all characters greater than eight are truncated and the file is not retrieved when you load RapidFile.

For example, entering "C:\DATA\FILE" returns the error message "File: "C:\DATA\.RPD" not found. Press any key to try again" when you load RapidFile. The status panel displays "C:\RAPID\RFDATA\C:\DATA\FILE." as the current database file. This assumes, of course, that your default drive is C and your default directory is C:\RAPID\RFDATA.

File:Modify structure and Changing Field Names

When you change a field name through **File:Modify structure**, all existing form layouts still reflect the old field name. Existing table and quick report layouts, however, reflect the new field name.

Field names displayed in form layouts are simply labels that default to the original field name. You can easily change these labels from **Layout:Design** pull-down menu.

Headers in Quick Reports

The default header for a quick report is "Report from < filename > .RPD on @date." This header is automatically created when you create a quick report layout. Therefore, if a database file has not yet been saved and given a name, the filename cannot appear in this header. Also, when a database file is renamed the default is not automatically updated to reflect the new filename.

To add a filename to a quick report header in an unsaved file or to change the filename in a quick report header for a renamed file, use the **Print:Print settings:Header** menu option.

HP LaserJet

When printing to the HP LaserJet, set the lines per page to 60 to avoid page creep. If you use the HP LaserJet exclusively, change this setting in RF Custom. Otherwise, you change this setting in the **Print:Print Settings** pull-down menu.

Negative Numbers

RapidFile cannot display negative numbers formatted for **Field format:Written** display through **Layout:Design layout**. If you try to output a negative number in this field format, asterisks appear in the field display.

Records:Total fields

When you execute **Records:Total fields** or **Ctrl-T**, the totals that result temporarily become records in the database file. This includes a blank field between the last record and the first total. You cannot, however, edit these fields.

To remove them, execute **Records:Total fields** or **Ctrl-T** again.

Saving Files

When you save a database file, it is saved to the currently selected drive and directory, not to the drive and directory it was retrieved from. If, however, you change the drive or directory while in **File:DOS Access**, files are still saved to the drive and directory selected from within RapidFile if the **File:Save File** option is chosen. However, if the directory is changed but the drive is not and the file is saved first by pressing **Ctrl-Return**, it will be saved to the directory chosen in DOS.

Zero to the Zero Power

The result of 0^0 is an undefined value. RapidFile, however, returns the value 1. This is inconsistent with dBASE III PLUS, which returns a 0.

ZOOM (F4)

If you press **ZOOM (F4)** to enter the memo writer in a text field containing exactly 254 characters and you then turn a typestyle on, the error message "Current memo too large to fit in cell. Save this memo and link it to the cell? Yes No" is displayed.

This occurs even if no more text is added since RapidFile places two attribute bytes in the field; one to turn the typestyle on and one to turn it off.

Line and Box Drawing with the HP LaserJet

BY MERYL HOROWITZ

The MultiMate Advantage Line and Box Drawing function allows you to include lines and boxes within MultiMate documents. The only requirement is that you have MultiMate Advantage version 3.50 or higher and a printer that supports the IBM Extended Character Set. For HP LaserJet owners, it means using the correct font cartridge. There are nine cartridges to choose from, each varying in the number of available line and box drawing characters and print pitches. Table 1 lists the cartridge name, the PAT (Printer Action Table) file to use, the available print pitches, and the available line and box drawing characters.

HP Cartridge	PAT filename	Print Pitches	Line & Box Characters
"G"	LJGLINE	1	1, 3, 5
"H"	LJHLINE	1	1, 3, 5
"R"	LJETRFS	3, 4, 7, 8	1, 2, 3, 4, 5, 6
"T"	LJETUFS	1	1, 3, 5
"U"	LJETUFS *	2	1, 3, 5
"V"	LJETUFS *	2	1, 3, 5
"W"	LJETW *	1, 3	1, 3, 5
"X"	LJETX *	1, 3	1, 3, 5
"Y"	LJETY	1, 2, 3, 4, 5	1, 2, 3, 4, 5, 6

Table 1.

* Using these PATs requires either an alteration to the PAT file or some special instructions before using them for line and box drawing tasks. These special instructions are listed below.

When you press **Alt-E** to invoke the line and box drawing function, six different line and box drawing characters display at the bottom of your screen. Figure 1 shows these characters as they appear when printed. The number of line and box characters available to you depends on which font cartridge you are using (see Table 1). Changing the print pitch of a document changes the pitch of the text only; it does not change the type style of the line and box characters. The line and box characters always print in the type style shown in Figure 1.

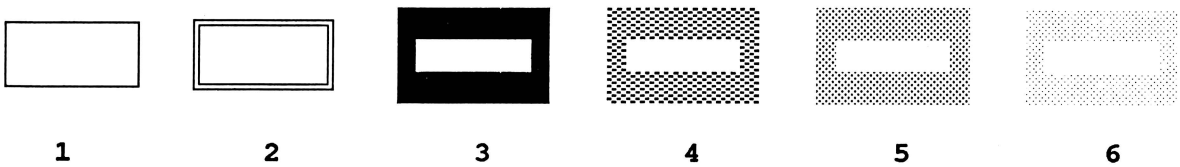


Figure 1.

For your convenience, Table 2 lists the nine different PATs and type styles that you will get with each print pitch. This table is a partial reprint from the table published in the April 1987 issue of ***TechNotes***, ("Using the HP LaserJet with MultiMate Advantage") which listed the print pitch attributes for all of the HP LaserJet font cartridges.

Print Pitches

HP Cartridge	PAT Filename	1	2	3	4	5	6	7	8	9
"G"	LJGLINE	12 cpi Portrait P.Elite Upright Medium	---	---	---	---	---	---	---	---
"H"	LJHLINE	10 cpi Portrait Courier Upright Medium	---	---	---	---	---	---	---	---
"R"	LJETRFS	---	---	10 cpi Portrait Pres.Bold Upright Bold	10 cpi Portrait Ltr.Gothic Upright Medium	---	---	10 cpi Landscape Pres.Bold Upright Bold	10 cpi Landscape Ltr.Gothic Upright Medium	---
"T"	LJETUFS	10 cpi Portrait Courier Uprt. Medium T.Line Box	---	---	---	---	---	---	---	---
"U"	LJETUFS	---	10 cpi Portrait Courier Uprt.Bold U LineBox	---	---	---	---	---	---	---
"V"	LJETUFS	---	10 cpi Portrait Courier Uprt.Bold U Line Box	---	---	---	---	---	---	---
"W"	LJETW	10 cpi Portrait Gothic 14 pt. Upright Medium	---	10 cpi Portrait OCR A 12 pt. Upright Medium	---	---	---	---	---	---
"X"	LJETX	10 cpi Portrait Gothic Upright Medium	---	10 cpi Portrait OCR B Upright Medium	---	---	---	---	---	---
"Y"	LJETY	10 cpi Portrait Courier Upright Medium	10 cpi Portrait Courier Upright Bold	10 cpi Portrait Courier Italic Medium	---	---	---	---	---	---

Table 2.

Using the "V" Cartridge

In order to print line and box characters using the "V" cartridge, you need to modify the LJETUFS PAT so that it will print the line and box characters in landscape mode. To make the changes, follow these steps:

1. Load the UTIL program and edit the PAT, LJETUFS.
2. Move to the Print Pitches area of the "Edit Printer Action Table" screen and change the BC token in pitch 1 to BD and press **F10** to save this in memory.
3. Move to the "Character Translations and Widths" screen and change the translation for character B1 from 9D7D95 to 9D7F95.
4. Press **F10** to save this in memory.
5. From the "Edit Printer Action Table" screen, select "Save PAT file (with new name)" and enter a new filename, such as LJETVLB.

When you wish to use the "V" cartridge to print line and box characters, specify LJETVLB (or whatever filename you chose) as the PAT filename on the "Print Parameters for Document" screen.

Using the "U" Cartridge

When printing line and box characters with the "U" cartridge, use the Character Width Translation Table, LJTUP2 in addition to the PAT, LJETUFS.

Using the "W" Cartridge

If you use pitch 3 with the "W" cartridge, you need to modify the CWT table and save it as a new file. These modifications instruct MultiMate Advantage to return to pitch 3 after printing line and box characters. To make the changes, follow these steps:

1. Load the UTIL program and edit the PAT, LJETW.
2. From the "Edit Printer Action Table" screen, select "Save Character Table as CWT File."
3. Enter LJETW3 as the CWT filename and then press **F10** to save it.
4. Press **Esc** to return to the "Edit Printer Tables" screen.
5. Select the "Character Width/Translation Tables" option and enter LJETW3 as the filename.

6. Each of the tokens for the line and box characters has a translation sequence that ends with the number 95. This instructs MultiMate to return to pitch 1 after printing the line or box character. To instruct MultiMate to return to pitch 3 instead of pitch 1 after printing line and box drawing characters, change the 95 to 97 in the following tokens: B1, B3, B4, BF, C0, C1, C2, C3, C4, C5, D9, DA, DB. The table below shows the default values of the translation sequences and what the new value should be.

Token	Default Value	New Value
B1	A47F95	A47F97
B3	A42E95	A42E97
B4	A43695	A43697
BF	A45495	A45497
C0	A44695	A44697
C1	A43895	A43897
C2	A43795	A43797
C3	A43595	A43597
C4	A42C95	A42C97
C5	A42F95	A42F97
D9	A44795	A44797
DA	A45295	A45297
DB	A44395	A44397

7. Press **F10** to save this modified CWT.

Using the "X" Cartridge

To use pitch 3 with the "X" cartridge, follow the instructions for "Using the 'W' cartridge" to edit the LJETX PAT. The only variation is that the translation sequences for the line and box characters are different. Use the chart below to change the last two numbers in each token from 95 to 97.

Token	Default Value	New Value
B1	C17F95	C17F97
B3	C12E95	C12E97
B4	C13695	C13697
BF	C15495	C15497
C0	C14695	C14697
C1	C13895	C13897
C2	C13795	C13797
C3	C13595	C13597
C4	C12C95	C12C97
C5	C12F95	C12F97
D9	C14795	C14797
DA	C15295	C15297
DB	C14395	C14397

Use this article as a reference when using the MultiMate Advantage line and box drawing characters with the HP LaserJet. Table 1 will help you in determining which HP LaserJet cartridge to use for the line and box drawing characters that you want. Table 2 lists the print attributes for the available pitches within each PAT. Refer to the table printed in "Using the HP LaserJet with MultiMate Advantage" for the print attributes for all of the HP font cartridges.

For instructions on how to include lines and boxes in your MultiMate documents, refer to the "Line and Box Drawing" section of the *Advanced Reference Manual.*

MULTIMATE TIPS

BY LIZ BURKE AND
MERYL HOROWITZ

Removing Underlining in Long Documents

On monochrome graphic monitors, underlined text in MultiMate Advantage displays in high intensity; so, depending on how you set the brightness on your console, you may not be able to tell what is underlined and what isn't. This can be a problem if you turn automatic underline on and then forget to turn it off— you may end up with a document that has much more underline than you want. If it isn't too much text, you can erase the underlining by pressing **Shift-_** on each character. As you can imagine, using this method on many pages of text is a slow process. So, for long documents of underlined text, you can delete the underlining by exporting your MultiMate Advantage document to an ASCII text file with the "Remove Underscore/Doublestrike" option set to Yes. Then you can import it as a MultiMate Advantage document and reinsert the underlining. The advantage of using this method is that you can strip your document of underlines fairly quickly. The possible disadvantage is that setting "Remove Underscore/Doublestrike" to Yes deletes underline, bold, shadow, and strike-out print codes throughout the entire document.

The New ASCII Import Function

The new ASCII Import Function allows you to import an ASCII File directly into a MultiMate Advantage II document. This new function is a simplified form of the more sophisticated ASCII File Conversion Utility (UTIL) and is intended to provide you with a quick and direct method for bringing information from an ASCII File into your MultiMate Advantage II documents.

While editing a MultiMate Advantage II document, you can access this function through the new pull-down menu system (**Alt-L** then **Ctrl-S**), or by pressing **Ctrl-6**.

Once you invoke the import function, MultiMate Advantage II prompts you to enter the name of the ASCII file to import from. At this point, you can list the files in the current directory or enter the filename to import from. For example, if you enter *.TXT and then press **F6**, MultiMate Advantage II displays all files located in the default directory with the extension TXT. If you enter a filename (including the drive, path, and file extension, if necessary) and press **F10**, it loads the ASCII file into your document at the current cursor position.

This new ASCII Import function works only in a Edit/Create mode. It does not work in the modes Drop Down Insert, Footnotes Edit, Format Line Edit, or within any Highlight function such as Move, Copy, Delete, and so on. Additionally, it does not work in Information Handling Data Files or Library files.

The imported ASCII file has several characteristics that you should be aware of:

1. When you import an ASCII file, MultiMate Advantage II inserts a format line above the imported data. The format line is set to 156 characters so that the original format of the ASCII file is maintained as much as possible.
2. In order to keep the import function as simple and as fast as possible, any special print attributes such as underlining, bold, shadow, and so on, that exist in the ASCII file are stripped when brought into the MultiMate Advantage II document.
3. Tab characters in the ASCII file convert to MultiMate tabs. Linefeed, carriage return/linefeed, and carriage return characters are brought into MultiMate Advantage II as carriage returns. Formfeed characters convert to page breaks.
4. Pitch symbols, footer symbols, and line and box drawing characters do not import into MultiMate Advantage II documents through this function.

Merge Printing from dBASE Files

The dBASE Merge function allows you to merge print using a dBASE II, dBASE III, or dBASE III PLUS database file (.DBF) as a Merge Data File. Your define block has to specify only the file type DBASE, as this example shows:

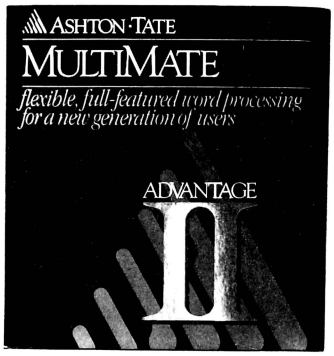
```
DOCUMENT: mergetest              ‖PAGE:   1‖LINE:   1‖COL:   1‖
|1--»----»----»-----------------------------------------------------------«———
├DEFINE├«
FILE TYPE DBASE«
├END DEFINE├«
«
June 1, 1987«
«
├NAME├«
├ADDRESS├«
├CITYSTATE├  ├ZIP├«
«
This is an example of a merge letter in Multimate Advantage II.  Notice
that no effort is necessary to set up the direct merge from dBASE files.
```

MultiMate Advantage II determines the field names, field sizes, and field types that make up your dBASE database file.

When merge printing from dBASE database files, always capitalize the Merge Item Names in your primary document because dBASE II, III, and III PLUS store field names in upper case.

Most of the data prints as it appears in the dBASE database file. However, there are some variations:

1. You cannot merge print from dBASE III memo fields.
2. If you are merge printing a column of numbers from a dBASE file, the decimal points may not align properly. If this happens, place a decimal tab before the numeric Merge Item Names in the primary document.
3. dBASE dates are formatted according to the Print Date option setting on the "System Defaults" screen.



ANNOUNCEMENTS

MultiMate Advantage II

BY MERYL HOROWITZ AND EILEEN JEFFERY

MultiMate Advantage II, the newest addition to the professional word processors in the Ashton-Tate family of software products, offers many new features that add power and flexibility to save you time and to allow you to customize your working environment more easily.

More Powerful

Each release of MultiMate has reflected changes requested by you, our users. Some powerful new features in MultiMate Advantage II include:

ASCII Import directly imports an ASCII file into your MultiMate Advantage II document without requiring you to go through the MultiMate Advantage II File Conversion Utility.

Binary File Downloading stores in the Printer Action Table up to 18 files that can be automatically downloaded to your printer (for example, Soft Font files for your Hewlett-Packard LaserJet printer).

Comments allow you to embed comments within a document that can be read by anyone viewing or editing the document on screen. You then have the option whether or not to include these in your printed document.

dBASE Merge allows you to merge data stored in a dBASE .DBF file with a MultiMate Advantage II merge document. This function further integrates MultiMate with the Ashton-Tate product line by allowing you to exchange information directly between the two products.

Preview Mode "prints" a MultiMate Advantage II document to the console instead of to the printer. This allows you to preview a document for header, footer, and footnote placement before actually sending the document to the printer.

Printif placed in a Define Block selectively prints records from a Merge Data File.

Sort reorders a section of text or numbers in ascending or descending order.

Undo Delete restores deleted text.

WordStar Conversion converts WordStar (.WST) files to the MultiMate Advantage II format through the MultiMate File Conversion Utility.

More Flexible

MultiMate Advantage II adapts to the way you use your word processor by offering additional user-controlled options such as:

Automatic Hyphenation: In addition to the existing User Selected Hyphenation option, you can now select Automatic Hyphenation to automatically hyphenate your MultiMate Advantage II document during Document Reorganization.

Document Mode: Now there are two ways to view your document—in the standard page-oriented mode or in Document Mode, which allows you to scroll through your document as continuous text without displaying any hard page breaks.

Hot Start: To execute MultiMate Advantage II and edit or create a document, you can go through the start-up screens (that is, Document Summary and Document Default screens) or "Hot Start" it to load MultiMate Advantage II and automatically create a new document or edit an existing document without going through the MultiMate start-up menus.

Pull-down Menus: MultiMate Advantage II now has pull-down menus that offer an alternate means of selecting and performing most editing and printing functions from within a document.

More User-Friendly

We at Ashton-Tate are committed to promoting a family of software products that continually strive to meet your needs. The integration of MultiMate products into the Ashton-Tate family can be seen in MultiMate Advantage II by:

- Its ease-of-use with pull-down menus similar to those found in dBASE III, dBASE III PLUS, Framework II, and RapidFile.
- Its almost transparent merge of dBASE II, dBASE III and dBASE III PLUS files with MultiMate Advantage II documents.
- Its revamped documentation, which is patterned after the Framework II documentation.

For more information on MultiMate Advantage II, please contact your nearest Ashton-Tate dealer or call the Ashton-Tate Customer Service Department at (203)522-2116.

Calendar of Ashton-Tate Training Courses

Ashton-Tate offers Standard Product Training Courses that provide overall training on the functionality of specific Ashton-Tate software.

The following Training Courses are open for attendance to you or any end-user within your company or both. These courses will be conducted simultaneously in all Ashton-Tate District Offices on the dates shown.

Please contact your nearest Ashton-Tate District Office, or call Ashton-Tate Training Services at 213/538-7738 for information on tuition and to reserve your place for training.

Class	Date
dBASE III PLUS	June 10–11
RapidFile	July 15–16
Graphics Overview	July 2–9
MultiMate Advantage II	August 12–13
Framework II	August 26–27
dBASE III PLUS	September 17–18
MultiMate Advantage II	September 23–24
Local Area Network	October 7–8
dBASE III PLUS Programming	October 12–13
Framework II	November 12–13
Local Area Network	November 18–19
Graphics Overview	December 2
MultiMate Advantage II	December 16–17

ANOMALIES

DEFINITION OF ANOMALY

We use the word "anomaly" to mean any one of the following:

- INCORRECT COMMAND (or function) IMPLEMENTATION. A command or function does not execute as specified.
- INCOMPLETE COMMAND (or function) IMPLEMENTATION. A command or function does not execute as specified in every use of the command or function.
- DATA OR LOGICAL BOUNDARY LIMITATION. A command or function cannot perform an operation at a specified data or logical boundary.

NOTE: Ashton-Tate reserves the right to withhold any further information that may reveal proprietary or trade secret information.

dBASE III PLUS

ACCEPT TO

In dBASE III PLUS, ACCEPT TO trims the character string entered of trailing blanks. Attempting to assign spaces to a memory variable with ACCEPT TO creates a null value.

dBASE III correctly accepted spaces with ACCEPT.

dBCODE

dBCODEing a ?? command may require a space between the question mark and the argument to execute properly. For example, dBCODEing the command

```
??REPLICATE(CHR(41), 5)
  ↑
  └──── No space.
```

produces the error message "Unbalanced parenthesis" when executed.

This problem does not occur when using the ? command.

DISPLAY STATUS

With binary files LOADed and SET STATUS ON, DISPLAY STATUS scrolls the top two lines of the SET values off the top of the screen. This occurs because DISPLAY STATUS does not account for the extra lines taken by the "Modules Loaded:" display.

This happens in both versions 1.0 and 1.1 of dBASE III PLUS.

IMPORT

In version 1.0 of dBASE III PLUS, IMPORTing a PFS file when a database file with the same name already exists and SET SAFETY is ON, answering "N" to the prompt "< Filename >.DBF already exists, overwrite it? (Y/N)" does not close the PFS file. Further attempts to access the PFS file produce the error message "File is already open."

To close the PFS file, you must QUIT dBASE III PLUS.

Read-only files

In single-user dBASE III PLUS, attempting to USE a database file whose attribute is read-only returns the error message "File is not accessible." The file is not opened and cannot be read.

Using MODIFY COMMAND to edit a read-only file does not produce an error message and the file is opened for editing. The modified file is saved correctly but the initial read-only file is renamed with a .BAK extension. Subsequent edits of the file are saved to a temporary file with the .$$$ extension as the read-only .BAK file cannot be erased or overwritten.

This occurs in both versions 1.0 and 1.1 of dBASE III PLUS.

RETRY

RETRY re-executes only the last physical line and not the entire command line when a command is broken with semicolons. For example, running the routine:

```
* ---Test.PRG
ON ERROR DO Test_err
? "This line ends with a semi-colon and " +;
  "this line is not terminated with a quote.
* EOP Test.PRG

* ---Test_err.PRG
? MESSAGE(), ERROR()
RETRY
* EOP Test_err.PRG
```

initially returns the correct error message "Unterminated string." However, RETRY executes only the second line of the ? statement and returns the error message "*** Unrecognized command verb" on subsequent iterations.

This occurs in both versions 1.0 and 1.1 of dBASE III PLUS.

Rounding

dBASE III PLUS occasionally rounds numbers ending in the digit five incorrectly. Rounding the number five should increment the number to the left of the five and drop the five, but occasionally the five is dropped and the number to the left is not incremented. This anomaly can occur wherever dBASE III PLUS is forced to round a number and not just when you use the ROUND() function.

For example, if you have a number with many decimal digits but SET DECIMALS is 2 and SET FIXED is ON, dBASE III PLUS is forced to round the number to two digits and the resulting value may be less than the initial number.

The following is a list of commands for which dBASE III PLUS rounds the results if they are numeric:

```
?/??
@...SAY/GET...PICTURE
DISPLAY/LIST
REPLACE
REPORT FORM
SET FIXED ON
STORE
SUM
TOTAL
VAL()
```

This can occur in both versions 1.0 and 1.1 of dBASE III PLUS.

SET CATALOG TO

On page 3-18 of *Using dBASE III PLUS* in version 1.0, the list of commands Catalog queries apply incorrectly includes SELECT. However, issuing

```
SELECT ?
```

with a Catalog open, returns the error message "Alias not found."

Also, APPEND FROM ? is incorrectly excluded from this list.

SET COLOR TO

It is not documented that entering a space (ASCII 32) as a parameter of SET COLOR TO sets the color to black. This is identical to the color option "N." For example, the commands

```
SET COLOR TO N/R,G/Y,RB
```

and

```
SET COLOR TO /R,G/Y,RB
             ↑
             └── There is a space here.
```

both set the foreground color to black on red. This applies to the COLOR keyword of CONFIG.DB, as well.

See the December 1986 issue of ***TechNotes***, page 41 for further information.

SET INDEX TO ?

Choosing an index file from the SET INDEX TO ? menu with no active Catalog produces the error message "Syntax error" when you exit the menu. Instead, SET INDEX TO ? without an active Catalog should display the error message "Data Catalog has not been established."

On page 5-181 of the version 1.0 *Using dBASE III PLUS*, SET INDEX TO ? is described as activating a menu of all index files for the active database file. The description should state that an active Catalog is required for this to occur.

This happens in both versions 1.0 and 1.1 of dBASE III PLUS.

SET PATH TO

When you SET PATH TO a list that includes the root directory, MODIFY COMMAND returns the error message "File is not accessible" when you attempt to create a file in any directory other than the default. For example, the following commands produce the error:

```
SET PATH TO A:\;C:\dBASE
MODIFY COMMAND C:\DOS\Test.PRG
```

This occurs in both versions 1.0 and 1.1 of dBASE III PLUS.

USE

A database file can be opened in two workareas simultaneously by USEing it with an ALIAS in the first workarea and then USEing it specifying the path to the database file location in the second.

For example, the database file Temp.DBF is in the default directory C:\DBASE. The following commands execute without error and Test.DBF is in USE in both workarea 1 and 2.

```
* ---The same database file in USE in two workareas at the
* ---same time.
USE Test ALIAS Temp
SELECT 2
USE \DBASE\Test
```

This occurs in both versions 1.0 and 1.1 of dBASE III PLUS.

Framework II

@reset

In Framework II version 1.0, certain illegal region references were not flagged as errors when you used **@reset**. For example, **Create:Database**, label it Test, and input data as shown below.

Name	Id	Other
First	23	
Last	45	

Next, enter the following formula behind the first record in the field labeled Other:

```
@reset(Test.Id.First:Test.Id.Last)
```

After recalculating the formula, version 1.0 returns the value 23, even though it is an illegal region reference. Version 1.1 returns the error constant #NAME?.

@unhide

If the argument of **@unhide** is a spreadsheet cell reference, your computer will freeze.

This happens in both versions 1.0 and 1.1 of Framework II.

Emulation with @echo(#OFF)

If a macro turns **@echo** #OFF and then goes into Emulation before turning **@echo** #ON, returning to the desktop displays a blank screen.

This happens in both versions 1.0 and 1.1 of Framework II.

MultiMate Export

1. Framework II does not always export attributed text (bold, underline, and italics) to MultiMate correctly. Because the MultiMate export utility only processes a line up to the carriage return; if the carriage return has an attribute, the Framework II sequence to turn the attributes off occurs after the carriage return, and the export utility never processes these hidden characters.

   The work-around is to turn **Edit:Display Hidden Characters ON**, highlight attributed carriage returns, and turn **Words:Normal ON** before exporting.

   This happens in both versions 1.0 and 1.1 of Framework II.

2. Framework II does not export multiple frames highlighted with **EXTEND SELECT (F6)** to MultiMate correctly. The second and subsequent frames are only partially written as .DOC files and can be read by MultiMate. When you load these files, however, the message "ERROR: UNABLE TO LOAD ENTIRE PAGE." is displayed the first time they are loaded, and the documents contain no data. In addition, each DOCUMENT SUMMARY SCREENs displays the name of the first file as the document name, and each contains the number of pages equivalent to its position in the export.

   The work-around is to select and **Disk: Export: MultiMate** the frames individually.

   This happens in both versions 1.0 and 1.1 of Framework II.

3. The Framework II MultiMate export utility does not export ASCII characters above 127.

   The work-around is to manually embed the characters from within MultiMate.

   This happens in both versions 1.0 and 1.1 of Framework II.

Other Telecommunications Programs

Running Crosstalk after a Framework II Tcom session may cause the A drive, the C drive, or both to be inaccessible until the system is rebooted.

This happens in both versions 1.0 and 1.1 of Framework II.

View Pagination

If the default paragraph marker is on the line just above a page break, a space is inserted before the paragraph marker, causing it to wrap to the next line. This occurs the first time the frame is printed or viewed with **Frames:View Pagination ON**, and causes the document to become one line longer than it was before the pagination or print was performed. For example, if the default paragraph marker is at the end of line 54, **Lines per Page** is set to 66, and top and bottom margins are each set to 6; a one page document is initially displayed when **Frames:View Pagination** is selected. If a second **Frames:View Pagination** is performed, the document becomes two pages with the last line on the second page.

The work-around is to force the default paragraph marker down to a line by itself by highlighting it and pressing **Return** before viewing pagination.

This happens in both versions 1.0 and 1.1 of Framework II.

RapidFile

Cursor Shape on a Corona Portable

On a Corona portable, pressing **Ins** does not change the cursor from blinking rectangle to blinking underline. The insert/typeover function is toggled properly, but the cursor remains a blinking rectangle.

Disappearing Footers

When a memo that contains a footer is saved, the footer may appear to be deleted the next time the memo is retrieved. This occurs only if you retrieve the memo after exiting and re-entering RapidFile. The footer, however, is not actually deleted, and pressing **Return** or **Spacebar** on the **Print:Print settings:Footer** option redisplays the footer.

Fields without Field names

It is possible to create a field in RapidFile that has no field name. This happens when you choose the **File:Create:New file** or **File:Create:Additional** field pull-down menu options and enter spaces as the field name. Once this field is created you cannot add it to a new layout using **Ctrl-Ins** or the **Fields:Show field** option.

The work-around is to change the field name through the **File:Modify structure** option.

Statistics in Quick Report

If report statistics are entered in the 47th and 48th column of a Quick Report, they are deleted when the layout is modified in **Layout:Design layout**.

The work-around is to re-enter the report statistics for columns 47 and 48 after you modify the layout.

MultiMate

Merge Print/Information Handling Data File

When preparing to print an Information Handling data file, if you view or edit the template page, you must also view at least one record of the data file before printing. If you view the template page and then immediately try to print, MultiMate Advantage will return you to the main menu without printing.

Info Handling Data File Record Maximum

An Information Handling data file can have a maximum of 250 records. However, MultiMate Advantage allows you to create more than that without giving you an error message. When you attempt to create the 251th record, the entire file is corrupted.

ON-LINE SERVICES

BY STEVE SILVERWOOD

Do you have a modem? Do you subscribe to CompuServe or The Source? If so, Ashton-Tate offers technical support on all of these electronic information services. You can quickly access the kind of quality technical information that you have come to expect from ***TechNotes***, and you can do it through your computer system. To take advantage of this progressive support method, you must subscribe to one of the services listed below. There are no additional charges for using our on-line support services, apart from the normal connect-time fees for each service. You can get specific subscription information by calling one of the following numbers:

Service	Subscriber Information	Command to find Ashton-Tate
CompuServe	(800) 848-8199	GO ASHTON
The Source	(800) 336-3366	ATSIG

To give you an idea of support that is happening on-line, the following are regular excerpts from the Ashton-Tate Forum on CompuServe. This monthly material comes from messages our Forum users have posted on the bulletin board. All message headers have been removed to protect the identities of our users. Every other message within a single conversation topic has been italicized to enhance readability.

ATO/ZAPCIS

If you use CompuServe often and would like to increase your efficiency (or just decrease your connect-time), you might want to try either Zapcis or ATO. These excellent shareware telecommunications programs are dedicated to sending and receiving information to and from CompuServe.

Basically, both Zapcis and ATO facilitate the reading of messages (your personal messages, all messages or combinations). The programs retrieve all the specified messages and disconnect (if you so specify). You can then read all the messages and compose responses to the ones you choose. Then whenever you like, you can have either program reconnect with Compuserve and post your responses.

Zapcis also makes it possible to receive the entire file catalog for various forum's Data Libraries, tag the files you'd like to download, and then leave the computer to do a batch download when it's convenient. This automation alone could save a great deal of connect time—no more searching around in Data Libraries, looking for the file you need.

Both programs are shareware, meaning that the authors (in no way affiliated with Ashton-Tate) expect some sort of remuneration if you like and use their programs. Explanations of the amount and how to pay are listed with the downloaded program files.

To get a copy of either Zapcis or ATO, GO IMCOM from the FUNCTION ! prompt. ATO has its own Data Library (DL 1), and Zapcis can be found in DL3. In either case, you'll need a copy of ARC.EXE (found in Ashton-Tate's DL0 as well as in the IBMCOM data library). This program is used to compress or uncompress files for uploading. In your case, you'll need to extract the files from the downloaded package. To do so, type

ARC E <filename>.

---

dBASE III PLUS

I am trying to APPEND a DELIMITED file. When I enter APPEND FROM <myfile> DELIMITED, I get the error message "File does not exist," yet the file entry displays when I run the DOS DIR command. What am I doing wrong?

If you the extension is not .TXT, you must type the filename with its extension. If the file has no extension, type the filename followed by two periods.

Thanks! My problem was simply that I did not specify an extension at all. When I renamed the file with an extension, everything worked fine.

.....

Do you know of any way to get dBASE III PLUS to replace numbers with words in a printout. Specifically, I'm programming a check-writing system and need to print the amount of the check in numbers and again in words.

There are several programs available to do this. First, there is an assembly language program in the March 1987 issue of ***TechNotes*** *on page 17. Second, on the Sample Programs and Utilities Disk there is a dBASE III PLUS program called Numwords.PRG. Third, there is a dBASE II program called Amt2word.PRG in the* Advanced Programmers' Guide. *Lastly, there is an explanation of this technique beginning on page 150 of* dBASE III PLUS Programming, Tips and Techniques *by Prague and Hammitt.*

.....

How can I use the current system date in a date field? I need to be able to SORT records by date of entry. To manually enter the date into each record is too time consuming.

Also, is there a macro I can use to enter sequential code numbers in specific fields of records that were previously entered? I need to select certain records to input a code number. Editing each record individually takes too long.

The answer to your first question is quite simple. After you APPEND a record, execute the following command:

REPLACE <field> WITH DATE()

As to your second question, create a program that selects a particular record and REPLACEs < field > WITH < code number >, where the code number increments with each record you select. For example,

```
* Program ...:   Serial.PRG
* Note(s) ...:   Serializes a set of records that match a
*           condition with consecutive numbers.
*
cntnue = .T.
serial =  1
*
* ---Numeric value, change 1 to whatever the starting serial
* ---number is to be.
DO WHILE cntnue .AND. .NOT. EOF()
   *
   * ---Place statements to select the desired record, and
   * ---display the data on the screen here (Use @..SAY
   * ---statements).
   @ 24, 1 SAY ''Serialize this record? '';
           GET choice
   READ
   IF choice
      REPLACE Field WITH serial
      serial = serial + 1
   ENDIF
   @ 24, 1 CLEAR TO 24,79
   @ 24, 1 SAY ''Select another record? '';
           GET cntnue;
           PICTURE ''Y''
   READ
   SKIP
ENDDO
* EOP Serial.PRG
```

I tried using the REPLACE command in my program and it does change the date field value to the system date after the record is inserted in the database. My problem is that I am using a .FMT file to input data so the date field is still blank. What I would really like to do is get the system date to appear on the screen when I issue APPEND and have the cursor move to the next entry field so the data entry person cannot change the date.

Instead of a GET statement for that field, simply use a command like:

@ < row >, < column > SAY DATE()

that displays the result of the DATE() function where you specify. Since it's a SAY command and not a GET, you cannot EDIT it. After the record is APPENDed, do a REPLACE for the date field before moving on to the next record to be added.

Your solution to my date field problem works great! Many thanks.

.....

I am writing my own print program and I wish to use the field names as headers. If the field is character, I can determine the size of the print width with LEN(FIELD(< field number >)). My problem is simply that if the field is numeric, LEN() fails. This leads to my two questions:

1. How do I determine the field type?

2. How do I find the default print size for numeric fields?

Many thanks for any help you can provide.

Use the TYPE() function to determine the type of field or variable you're using. For example:

```
x = 100
? TYPE("x")
```

will return "N" because x is a numeric variable.

The default print size for numeric fields is the width of the field. Unfortunately, when you use STR() without a length parameter to convert a numeric to character, the length of the output string is always 10 no matter what the length of the field is. This limits what you can do with FIELD() to create data independent applications. You can, however, create data independent applications with an Extended Structure file. An Extended Structure file, if you are not familiar with the term, is a database file that has as its fields the attributes of a database file structure (name, type, width, and decimals). You create this database file with the command, COPY STRUCTURE EXTENDED TO < extended filename > . USE it in the same way you would any other database file. Once it is open, you can navigate to the field you want and access all of the field attributes. It is really quite that simple.

To see examples of this technique take a look of some of the sample programs in ***TechNotes****. We wrote many using Extended Structure files in order to make them as widely applicable as possible. For example, see p. 28 of May 1986 and p. 52 of March 1987.*

.....

I am using dBASE III in conjunction with the Leading Edge Word Processor and cannot get the combination to mailmerge properly. The Leading Edge manual lists the appropriate commands to prepare a dBASE III database file for use as a Merge Data File, i.e., converting it to a delimited file. The problem seems to lie in getting SDF and DELIMITED to work on the same command line. Each time I try this:

```
COPY TO < filename > SDF DELIMITED
```

I get an error message. If I use only DELIMITED, the command works, but when I use the resulting Merge Data File with the form letter created with the word processor all the fields merged from the database file are surrounded by quotation marks.

The Leading Edge Word processor manual incorrectly states the syntax of the COPY TO command. SDF and DELIMITED are mutually exclusive. SDF copies the data out in a columnar format, with each record separated by a fixed width, padded with spaces (so it looks like a spreadsheet). DELIMITED puts quotations around each field, and the default separator is a comma. There is no way that I know of to suppress the quotes when COPYing DELIMITED. You can change the default delimiter by adding WITH < character > to the COPY TO DELIMITED command line. If you absolutely need to export without delimiters but must have commas as separators, you can use the program Cpdelim.PRG on page 28 of the May 1986 issue of ***TechNotes*** *to accomplish this.*

The following is a program that marks duplicate records as deleted for further processing. However, every time I run it, the ERASE command keeps returning the error message "file not found." I can ERASE the file from the dot prompt but can't understand why it won't work here. Any suggestions?

```
PARAMETERS database, fieldname
SET TALK ON
USE &database
temp = TRIM(fieldname) + ".NDX"
ERASE TRIM(fieldname) + ".NDX"
INDEX ON &fieldname TO &fieldname
GO TOP
DO WHILE .NOT. EOF()
    temp = fieldname
    SKIP
    IF temp = fieldname
       DELETE
    ENDIF
ENDDO
RETURN
```

Since you have temp = TRIM(filename) + ".NDX", why not have a command like ERASE &temp?

I could ERASE &temp, but I would like to know why ERASE fieldname + ".NDX" doesn't work.

If you ERASE TRIM(fieldname) dBASE III PLUS looks for a file called TRIM(fieldname), literally. Of course this file does not exist and so you get the error message "File does not exist."

.....

Framework II

Is it possible to import Lotus 1-2-3 Release 2.0 worksheets directly into Framework II?

Framework II can read 1-2-3 Release 1A files, but not Release 2.0 files. Convert the Release 2.0 files to Release 1A by using 1-2-3's conversion utilities, then use Framework II to read the converted spreadsheets.

.....

I am having a problem dealing with one particular thing in Framework II. How can I remove the "/p" shown by **Frames:Reveal Type** indicating local print menu settings are active for that frame? I have gone through and deleted all of the settings for these frames on the **Print:Format Options** and **Print:Control Options** submenus. This still did not remove the local settings indicator. The only way I have found to work around this is to **COPY (F8)** the entire contents to a new frame.

Another question I have is what exactly is meant by the setting in Setup referring to the maximum lines stranded? Should this really read Minimum instead of Maximum? Are these lines stranded on the page before or after the break? There is no explanation in the documentation.

You have correctly deduced that there is no way to remove the "/p" once a frame has had some attributes changed. The only way is to ***COPY (F8)*** *the original contents to a new frame.*

As to your second question, the "maximum lines stranded" is also called the "widow/orphan tolerance." This is a misnomer, since it refers to the minimum and not the maximum number of lines of a paragraph Framework II allows alone on a single page. For example, suppose "maximum lines stranded" is set to 3. When you print, any paragraph less than 6 lines is always kept on one page. This is because to split it would mean that less than 3 lines would be on one page or the other. For the sake of clarity, it could also be called "Minimum number of lines for a paragraph to be placed on current page."

.....

Does anyone know of an easy and simple way to bold frame labels in an outline without changing the attributes of the contents? The only thing I could think of was to create a macro that walks down the outline to change the label for each frame.

Your assumption is correct. To set the attribute of a frame label, enter Label editing, highlight the text, and then select the attribute from the ***Words*** *pull-down menu. To bold frame labels in an outline,* ***TechNotes*** *writers use a macro that does this:*

@performkeys(" "&"{F6}{Ctrl-End}{Ctrl-W}B{Return}{Dnarrow}")

To set an attribute to the labels of more than one consecutive frame in an outline, press ***Ctrl****, the number times you want the macro to execute, and the* ***Alt*** *key you set macro to. This effectively automates your task.*

Thanks. I used your macro and it made life a little easier. Wouldn't it be consistent and more powerful to have outline mode operate on the frame labels only, just like **Print** does? Also, it would be nice if one could search and replace any attributed text.

I like the idea for search and replace of attributed text.

.....

Is there a way to underline a header in Framework II?

Yes, but you must use print formulas instead of the header and footer options of the ***Print Format Options*** *submenu. Basically, you place functions like* ***@hl, @hr, @fc****, etc., in the formula area of the frame you want to print, set the attribute on the header or footer string from the* ***Words*** *pull-down menu, and set* ***Print:Control Options:Recalc Formula during Printing*** *on. For example, to set a header left to bold place a formula like the following in the formula area of your frame:*

*@hl("****Bold String****")*

This technique works well if you are setting an attribute for a literal string, but does not work at all if you want to set an attribute to the result of a formula such as a page number (with ***@pn****). In this case, use the user-defined function @Attrib (see the March 1987 issue* ***TechNotes,*** *page 56) to set the attribute to the calculated string. As an example, suppose you wanted today's date underlined as the header right. The following formula using @Attrib will accomplish this:*

@hr(@Attrib(@date4(@today),2))

Hope this helps you.

.....

MultiMate

We are having problems merge printing documents on the Novell Network. We are using MultiMate 3.3 Series with the Novell NetWareR/86 version 2.01 operating. The problem is that we cannot get proper top-of-form.

Try not specifying the Sheetfeeder Action Table "Network" on the Print Parameter For Document Screen and set all bins to 0.

.....

I need some assistance in selecting the correct MultiMate package for use on the IBM Token-Ring Network. The advertising is not clear whether the IBM PC Network version is appropriate for this application. Your assistance would be appreciated.

MultiMate IBM PC Network version 3.60 has been certified on the IBM PC Token-Ring Network hardware, running the IBM PC Network Program software versions 1.1 or 1.12. One problem that we found during testing is that the network printer cannot be password protected or the user will get "Net Print Error." We do not have a work-around at this time.

.....

I am using MultiMate Advantage version 3.60B and the HP LaserJet™ printer with font cartridge B. What procedure is to be used within a document to use Times Roman, point 10; Times Roman, point 8; and Helvetica, point 14? How is one to set up the line length for proper centering of all three point sizes assuming the primary point size will be 10. May I use just one CWT for the above? Lastly, is there PAT support for the HP LaserJet Soft Fonts?

With the 3.60B version of MultiMate Advantage, margins and centering are determined by the default pitch on the Print Parameters For Document screen. If the default pitch on the Print Parameters For Document screen is pitch 2, 3, or 4 (PS 10 Pt. font), the suggested format line length would be 80. If the default pitch is 1 (PS 14 Pt. font), the suggested format line would be 60. If the default pitch is 5 (PS 8 Pt), the suggested format line would be 121.

To access proportional spacing in MultiMate Advantage with the B cartridge, it is required that the LJETBPS PAT file be used and that all the CWTs (LJETBPS1, LJETBPS2, LJETBPS3, LJETBPS4, and LJETBPS5) that coordinate with the pitches used in the document be copied into the MultiMate Advantage subdirectory. Use the MultiMate Advantage pitch command to access all of the above fonts within one document.

Owing to these requirements, it is not possible to use only one CWT for the specified document. It is not necessary to specify the name of a CWT on the Print Parameters For Document screen.

Yes, there is PAT and CWT support for the HP LaserJet Soft Fonts. At this time we support the Times Roman and Helvetica Soft Fonts. We are presently working on CWTs for the Prestige Elite and Letter Gothic Soft Fonts. Additional information on HP LasetJet Soft Font support can be found in DL4 in the file HPSOFT.DOC. Hope I have been of some assistance.

.....

CORRIGENDA

March 1987, page 13
Field Ditto During Data Entry

On page 13 of the March 1987 issue, there is mistake on the command line that calls the program Ditto.PRG. To correct, replace the following line:

DO Ditto WITH <number of fields>, <.DBF filename>

with

DO Ditto WITH <number of fields>, "<.DBF filename>"

March 1987, page 34,
Scroll-Rate

There is a critical error in the Scroll Rate macro on page 34. To correct and make the Scroll Rate operational, replace the subframes, Mainrate and Pager with the following versions:

```
; MainRate
;
@local(getrate),
@eraseprompt,
getrate := @inputline("How many lines to scroll?"),
@eraseprompt,
Rate    := @if(@iserr(@value(getrate)), 0, @value(getrate)),
@if(rate < 1,
    @list(@keyfilter({Ctrl-Uparrow}),
      @keyfilter({Ctrl-Dnarrow}),
      @return(0)
   )
),
@keyfilter({Ctrl-Uparrow}, Pager),
@keyfilter({Ctrl-Dnarrow}, Pager)
```

```
; Pager
;
@performkeys(@rept("{" & @mid(@keyname(@key), 7, 2) & "arrow}", Rate))
```

May 1987, page 3, Say2file.BIN

If you CALL Say2file.BIN within the Developer's Release, you will encounter an anomaly that makes it appear that Say2file.BIN does not work. To correct the problem, rename Say2file.BIN to a filename seven characters or less in length.

May 1987, page 32, The Power of Calculated Fields

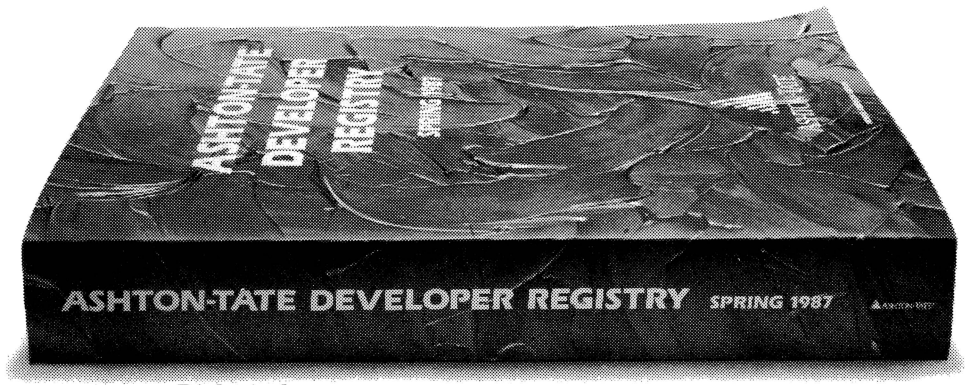
In the table on page 32, replace the Expression field for the fourth field, Send, with

```
IFSTR$ ((TODAY() > Next - 10 .AND. TODAY () < = Next),
"Send bill", "")
```

Also, replace the Expression field for the fifth field, Transit, with

```
IFSTR$((TODAY() > Next .AND. TODAY() < Next + 30),
"Transit", "")
```

If you need solutions, we're divulging sources.



Right in here.

It's a comprehensive listing of over 1,000 products and services from software developers and consultants nationwide. Cross-referenced and indexed so that you can easily find the specific solutions to meet your business software needs. And get the most out of Ashton-Tate® products like dBASE,® Framework,® RapidFile,™ and MultiMate.®

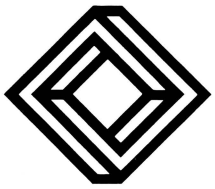
No matter what application you're looking for, you're likely to find it here. And to make sure you do, we've listed sources not only by business category, but also by product name, geographic locale, even area code. So you can't miss.

It's *the* sourcebook of Ashton-Tate software specialists, and you'll want to have it at your fingertips.

The *Developer Registry.* $24.95 at your local bookstore, or to order by phone call (800) 437-4329 Ext. 2208.

ASHTON·TATE®

In Colorado call (303) 799-4900, Ext. 2208. Trademarks/owner: dBASE, Framework, MultiMate, RapidFile, Ashton-Tate/Ashton-Tate Corporation. ©1987 Ashton-Tate Corporation. All rights reserved.



SUBSCRIBE TODAY TO

TechNotes

AND SAVE $22 OFF THE NEWSSTAND PRICE!

Get Ashton-Tate's hottest source of technical know-how at 30% off the newsstand price! Each monthly issue of **TechNotes** is filled with the most current, accurate technical information available on all dBASE, Framework, and RapidFile products. Subscribe today! Return this card today or call our toll-free numbers: **(800) 331-4164** (in California) or **(800) 828-2514** (residents of other states). In Canada, call **(619) 487-3212**. Newsstand price is $6 per copy (in U.S.A.).

☐ Bill me:
Amount enclosed:
☐ $50 (U.S.)
☐ $65 (Canada & Mexico)
☐ $75 (Other countries)
☐ New subscription
☐ Renewal

Form of Payment enclosed:
☐ Check (Make payable to Ashton-Tate in U.S. dollars drawn on a U.S. bank)
☐ Visa ☐ MasterCard
Expiration date:

Signature:

Name

Address

City State/Province

Zip/Postal Code Country

Phone Number

Which product do you use?
☐ dBASE II (D2) ☐ dBASE III (D3) ☐ dBASE III PLUS (DP)
☐ Framework (FW) ☐ RapidFile (RF)

Please allow 30 to 60 days for delivery of your first issue. 1000

READER SURVEY

We would like to make sure that ***TechNotes*** continues to serve the needs of our readership. Please take the time to complete this survey.

Which Ashton-Tate product do you use?

☐ dBASE II ☐ dBASE III/dBASE III PLUS ☐ dBASE III PLUS Lan Pack
☐ Framework II ☐ RapidFile

How long have you been using these product(s)? _____ years _____ months

How long have you been a subscriber to ***TechNotes***? _____

Rank in order of most usefulness (1-8):

_____ **Articles**	_____ **Sample Programs**	_____ **Letters**
_____ **Usage Tips**	_____ **Anomalies**	_____ **On-Line**
_____ **Corrigenda**	_____ **International**	**Services**

This article	**was/was not useful (because):**	**would have been better if:**
Save and Restore Partial Screens		
Snaked Column Reporting		
Using the *Framework II Developer's Toolkit* with Microsoft C		
Translating FRED Programs from One Language to Another		
Automating Applications Using Macros		

IF YOU'RE MOVING

Don't forget to tell us so that we can be sure your subscription to **TechNotes** and/or the Ashton-Tate Quarterly is uninterrupted. To process a change of address, we must have your previous mailing address (including the zip code) and your new address.

My new address is:
Name/Department
Company Name/Add'l Destination Information
Mailing Address
City/State Zip Country

Please affix your current mailing label(s) here. (If the label is not available, be sure to include your old address.)

Affix your current Ashton-Tate Quarterly mailing label here

Affix your current TechNotes mailing label here

To change your address on the Ashton-Tate Product Mailing List, necessary to receive upgrade and other important information about your Ashton-Tate product(s), you must send a separate change of address to: Ashton-Tate Customer Service, 20101 Hamilton Ave., Torrance, CA 90502-1319.

n

W

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 66 TORRANCE, CA
POSTAGE WILL BE PAID BY ADDRESSEE

TechNotes

Subscription Service
P.O. Box 3729
Escondido, CA 92025-0929

Place
Postage
Here

TechNotes

Reader Survey
Ashton-Tate Software Support Center
20101 Hamilton Avenue
Torrance, CA 90502-1319

Attention: Brenda Johnson-Grau

Place
Postage
Here

Ashton-Tate Periodicals
PO Box 3729
Escondido, CA 92025-0929

TechNotes
P.O. Box 3729
Escondido, CA 92025-0929

Address Correction Requested

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 324
GARDENA, CA